

# Data Mining

## Classification: Basic Concepts, Decision Trees, and Model Evaluation

---

---

### Lecture Notes for Chapter 4

Introduction to Data Mining

by

Tan, Steinbach, Kumar

# Classification: Definition

---

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

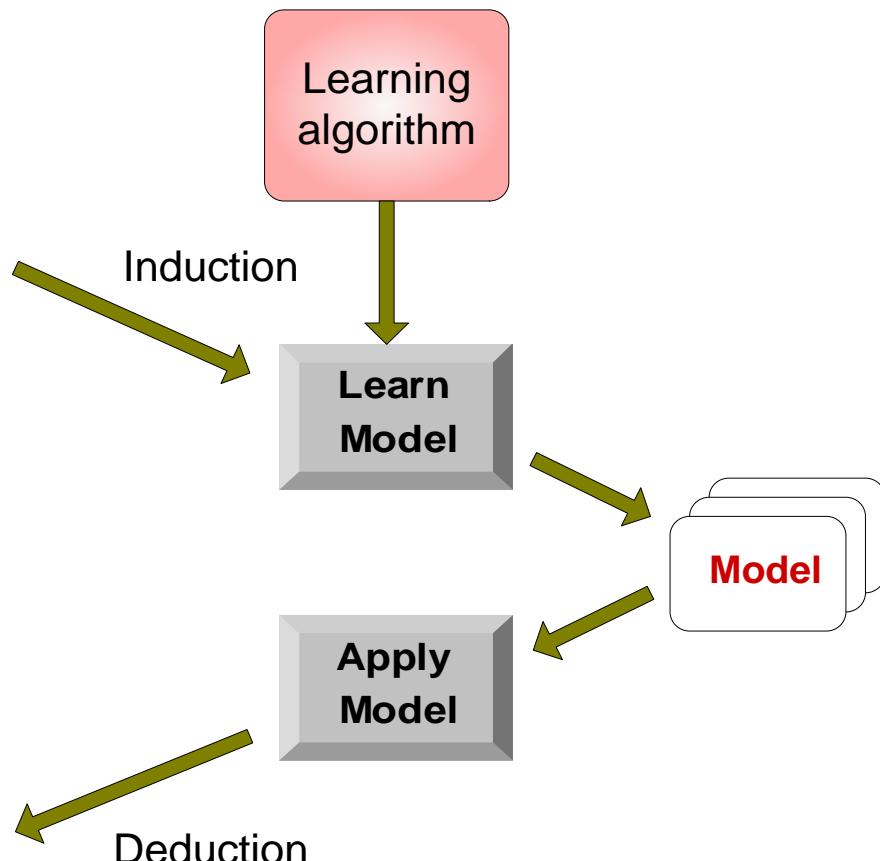
# Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# Examples of Classification Task

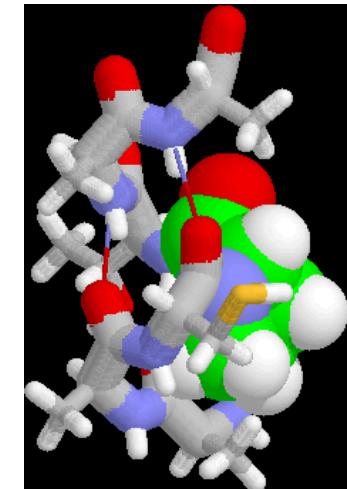
□ Predicting tumor cells as benign or malignant

□ Classifying credit card transactions  
as legitimate or fraudulent



□ Classifying secondary structures of protein  
as alpha-helix, beta-sheet, or random  
coil

□ Categorizing news stories as finance,  
weather, entertainment, sports, etc



# Classification Techniques

---

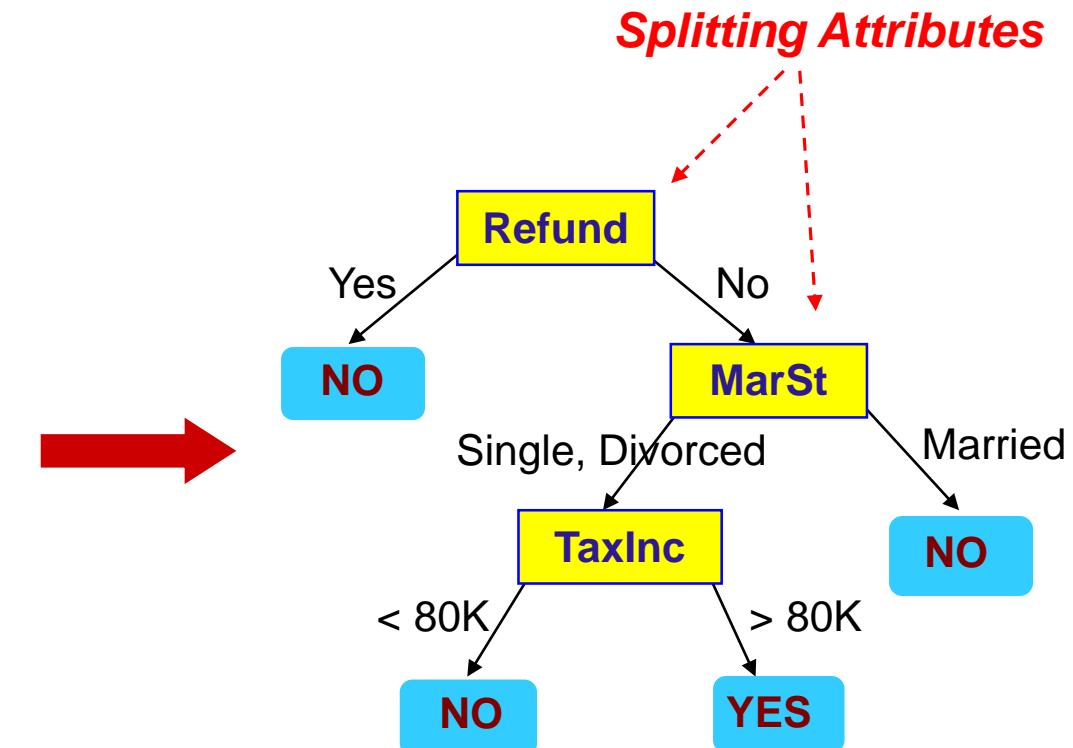
---

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

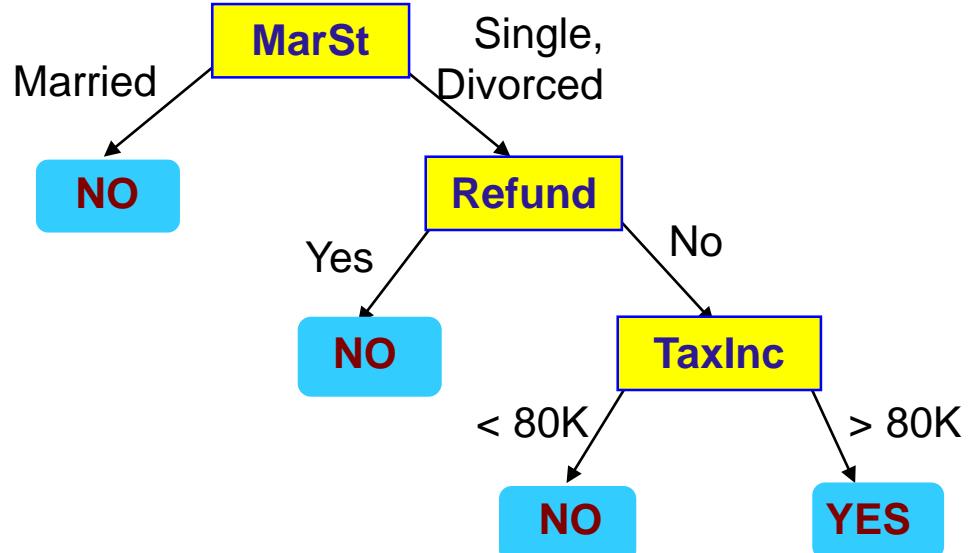
Training Data



Model: Decision Tree

# Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat	categorical categorical continuous class
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	



There could be more than one tree that fits the same data!

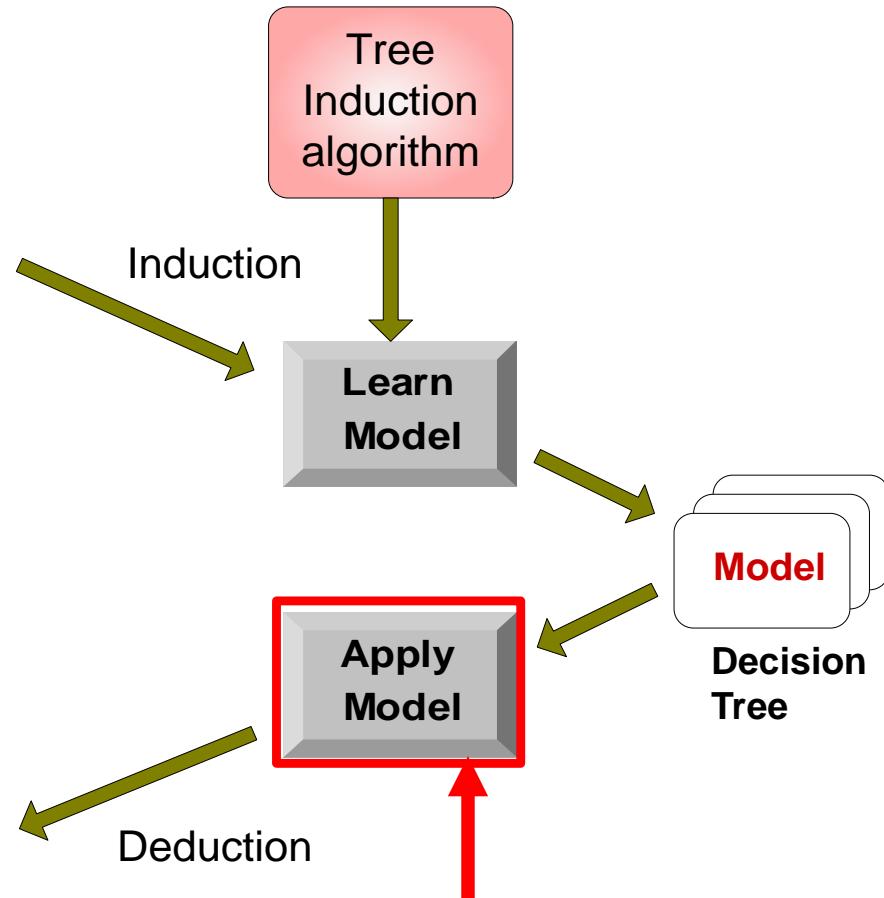
# Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

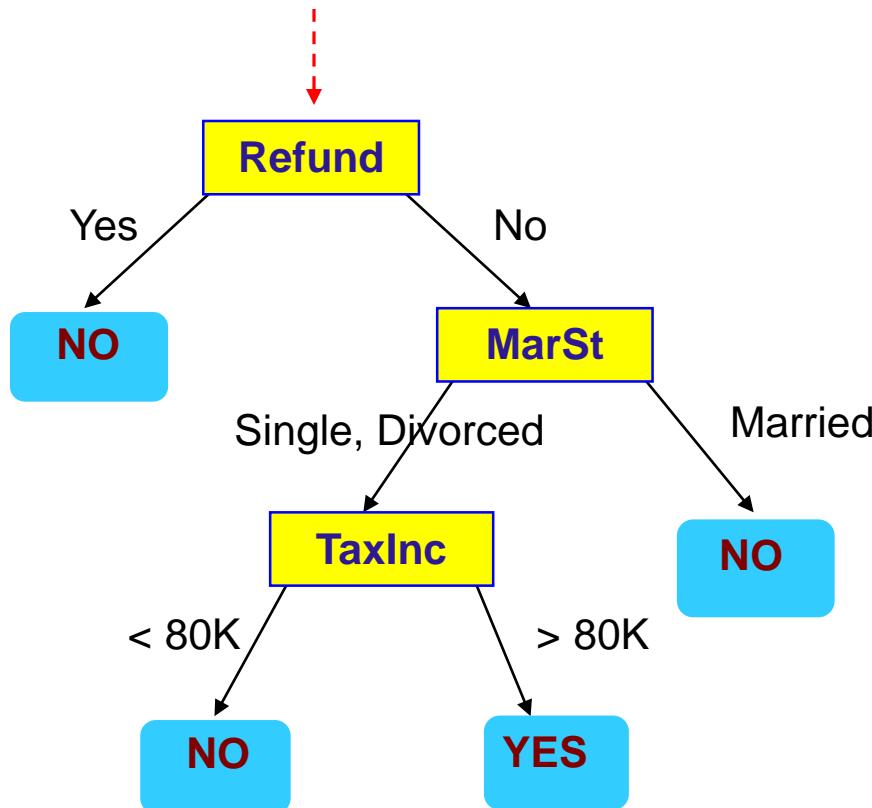
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# Apply Model to Test Data

Start from the root of tree.



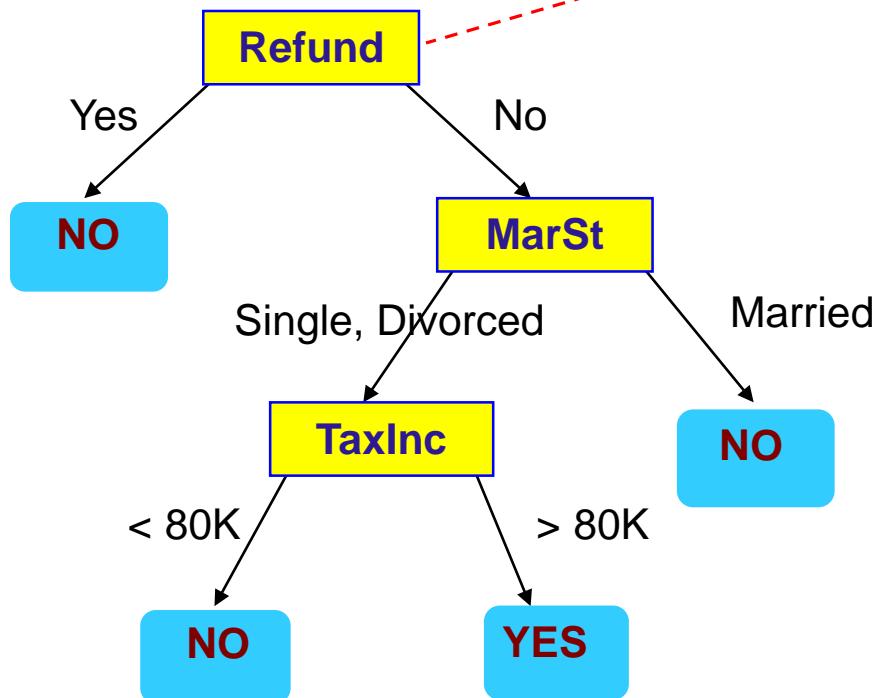
## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

# Apply Model to Test Data

Test Data

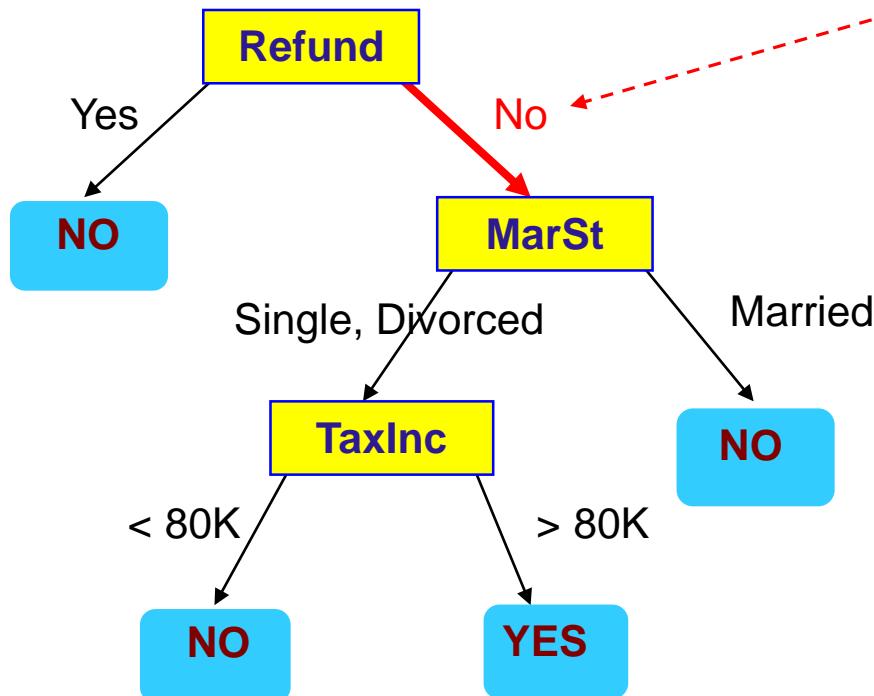
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

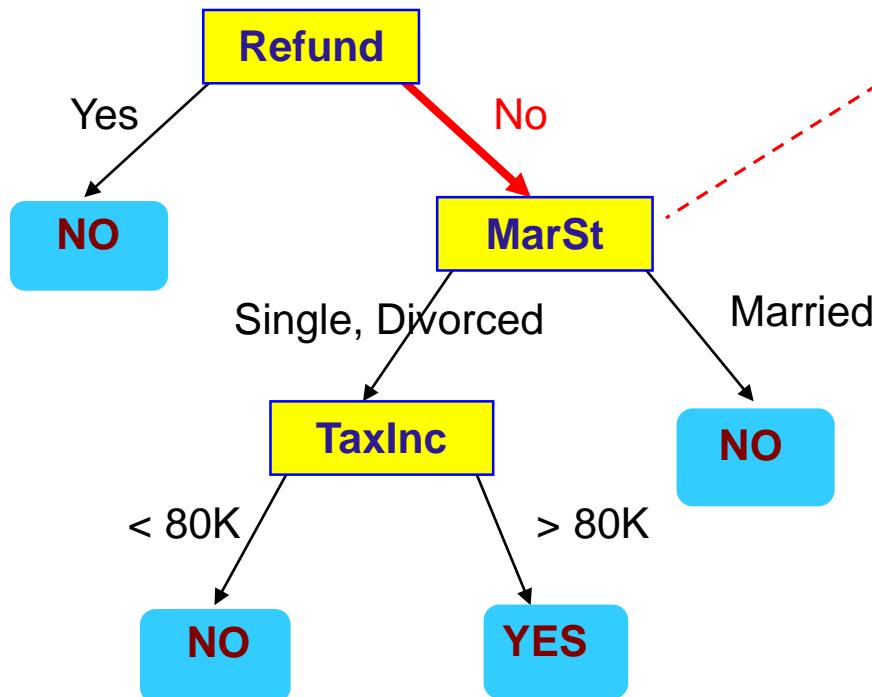
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

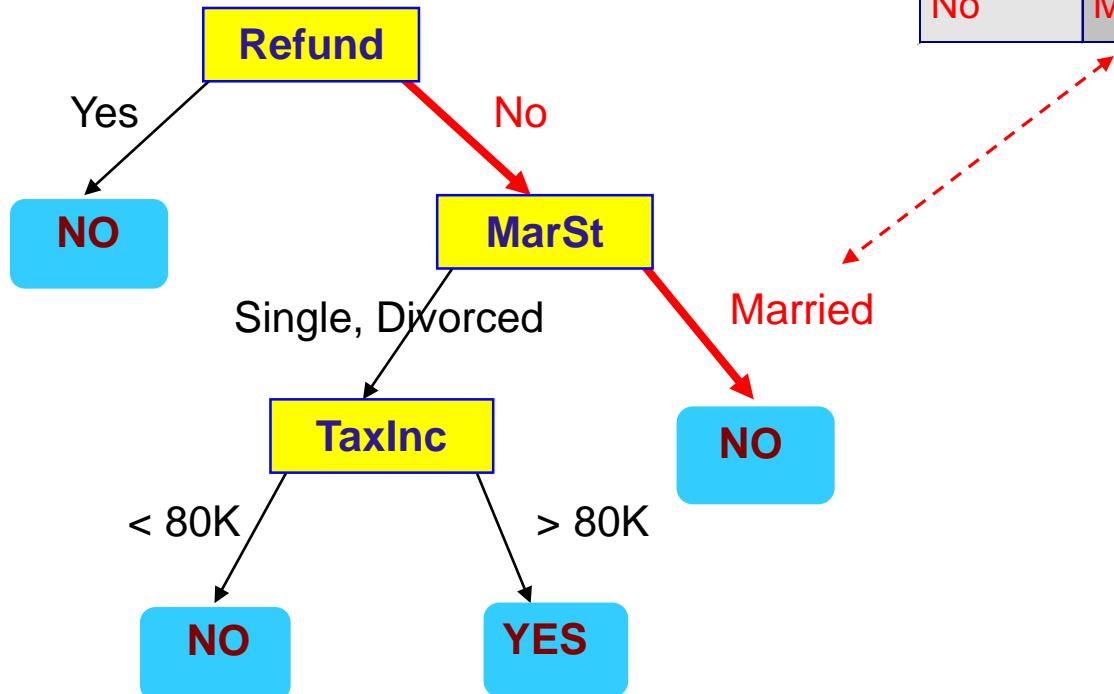
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

## Test Data

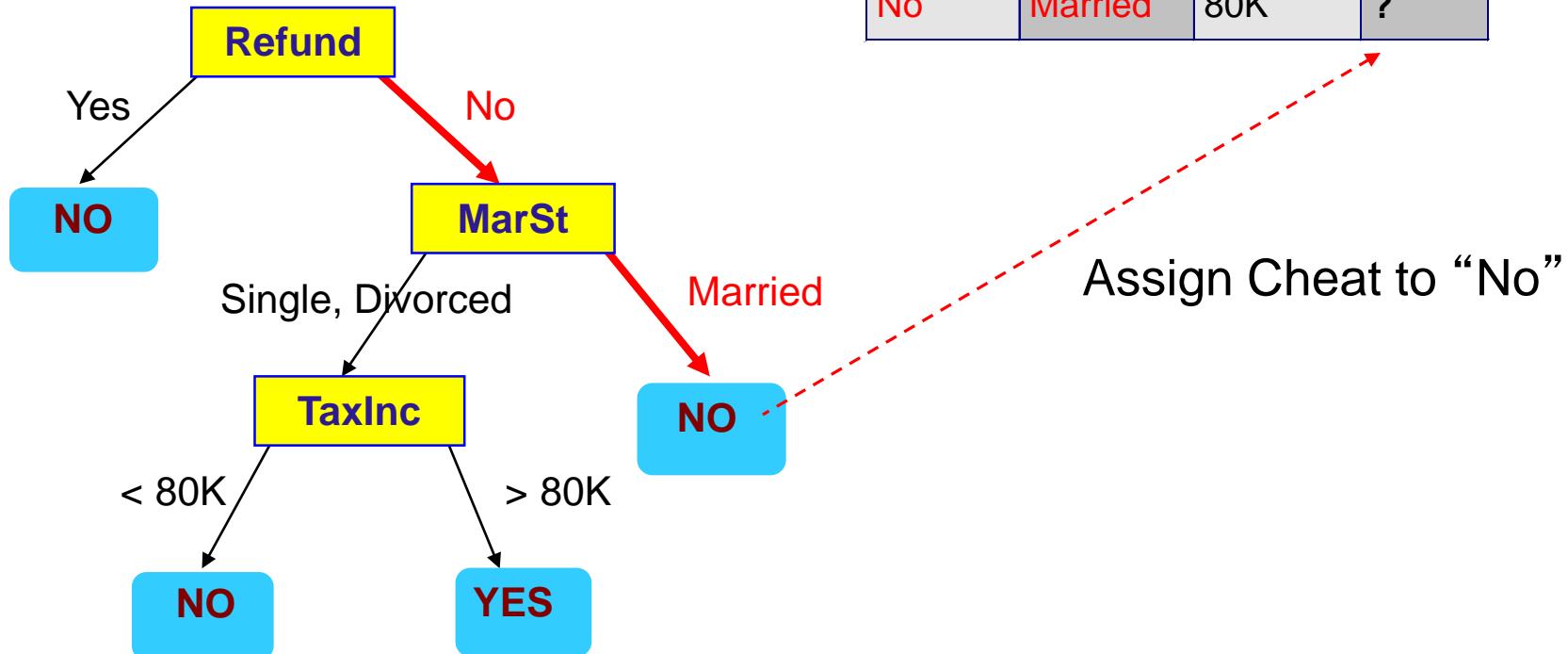
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



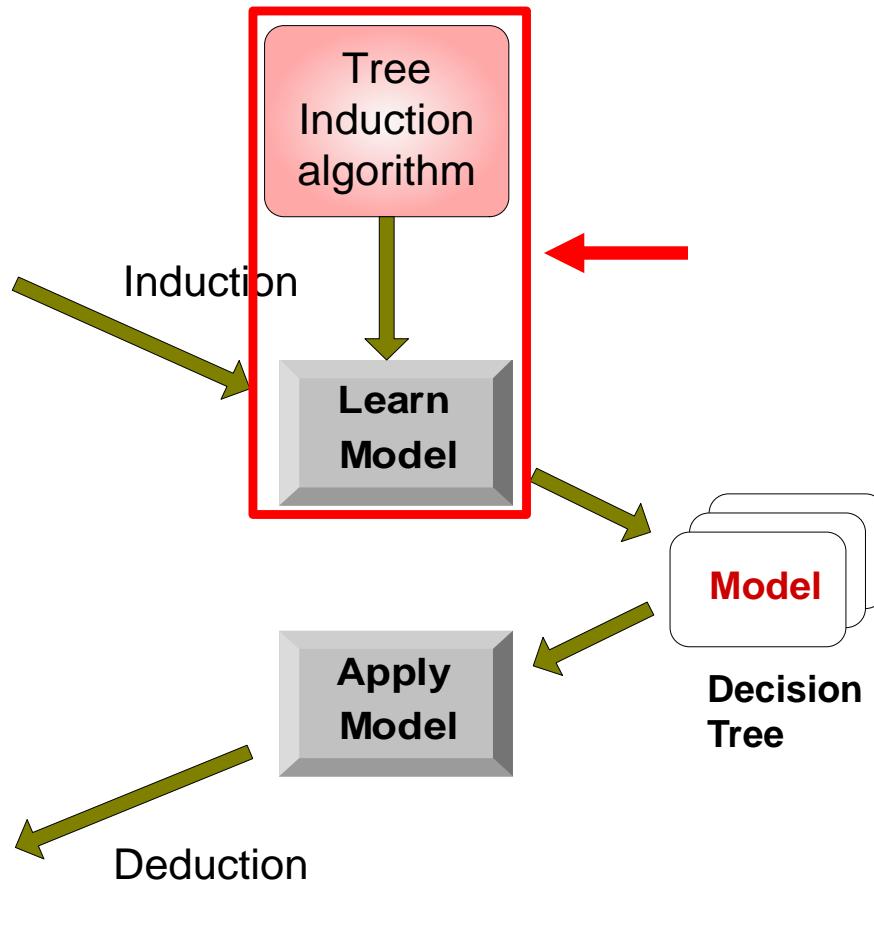
# Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# Decision Tree Induction

---

---

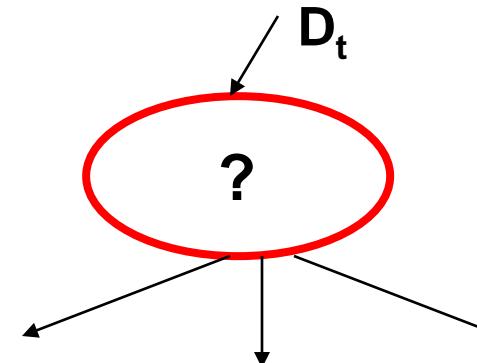
## □ Many Algorithms:

- Hunt's Algorithm (one of the earliest)
- CART
- ID3, C4.5
- SLIQ, SPRINT

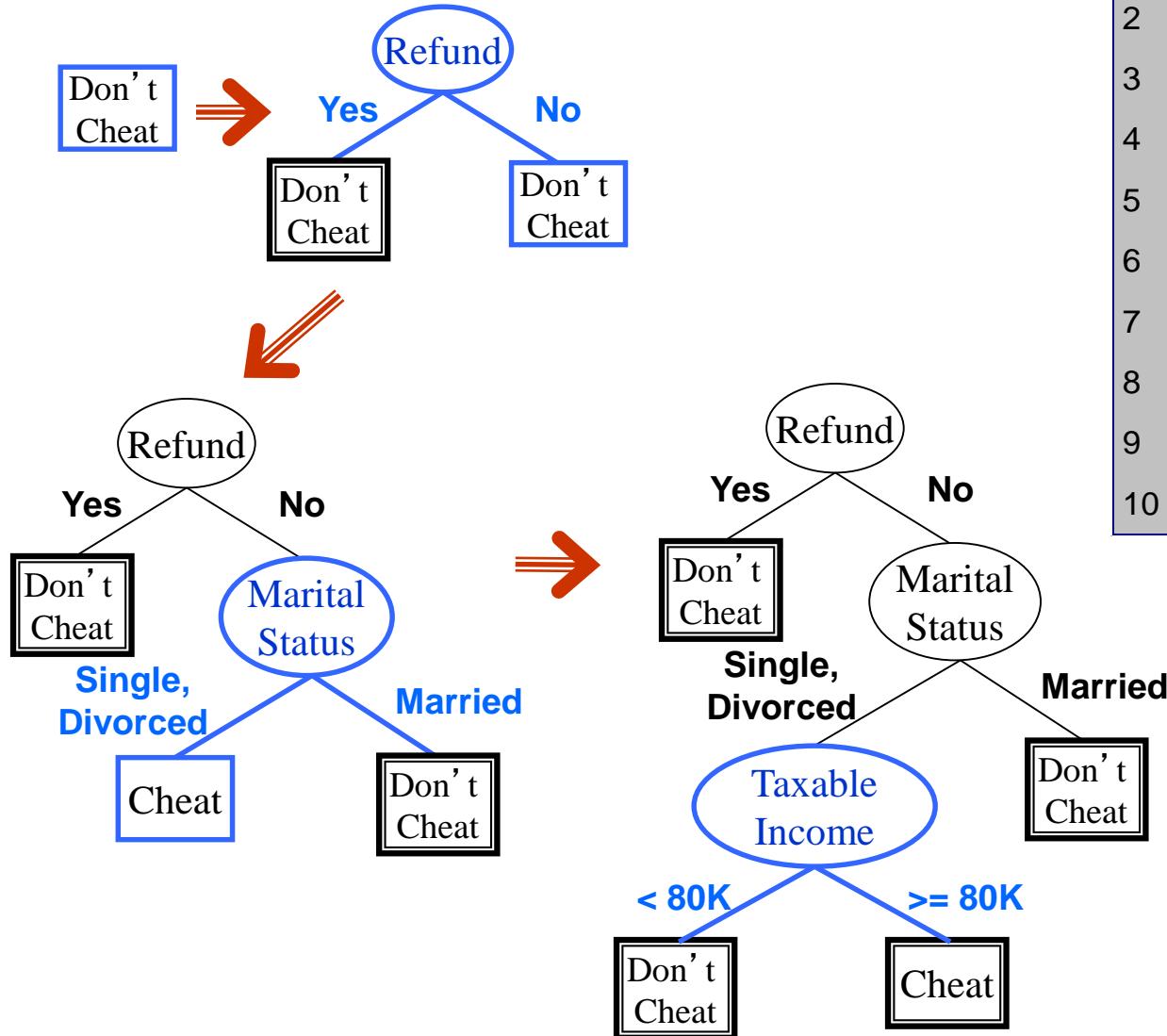
# General Structure of Hunt's Algorithm

- Let  $D_t$  be the set of training records that reach a node  $t$
- General Procedure:
  - If  $D_t$  contains records that belong the same class  $y_t$ , then  $t$  is a leaf node labeled as  $y_t$
  - If  $D_t$  is an empty set, then  $t$  is a leaf node labeled by the default class,  $y_d$
  - If  $D_t$  contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



# Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Tree Induction

---

---

## □ Greedy strategy.

- Split the records based on an attribute test that optimizes certain criterion.

## □ Issues

- Determine how to split the records
  - ◆ How to specify the attribute test condition?
  - ◆ How to determine the best split?
- Determine when to stop splitting

# Tree Induction

---

---

## □ Greedy strategy.

- Split the records based on an attribute test that optimizes certain criterion.

## □ Issues

- Determine how to split the records
  - ◆ How to specify the attribute test condition?
  - ◆ How to determine the best split?
- Determine when to stop splitting

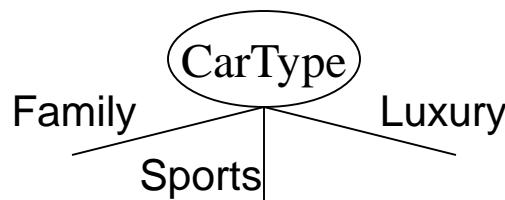
# How to Specify Test Condition?

---

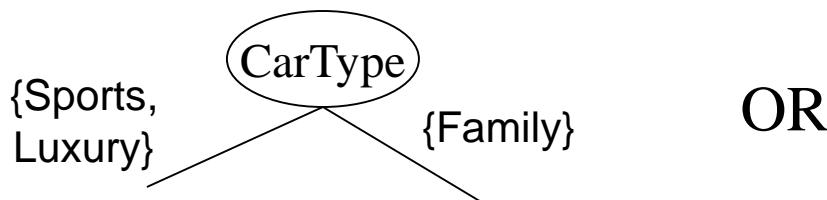
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
  
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

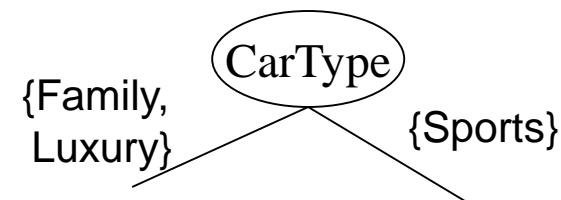
- **Multi-way split:** Use as many partitions as distinct values.



- **Binary split:** Divides values into two subsets.  
Need to find optimal partitioning.

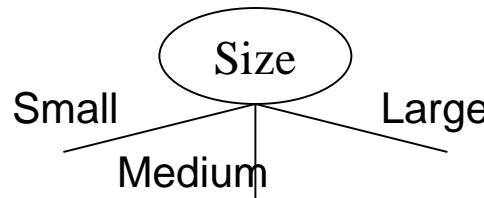


OR

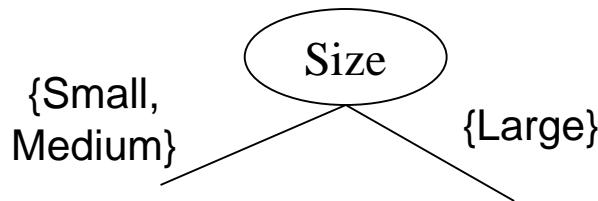


# Splitting Based on Ordinal Attributes

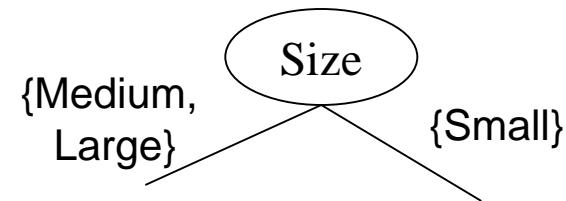
- **Multi-way split:** Use as many partitions as distinct values.



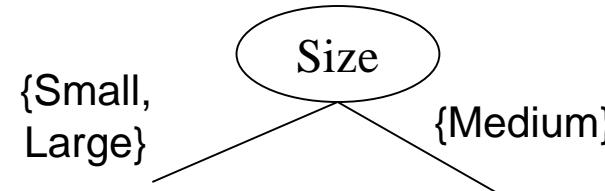
- **Binary split:** Divides values into two subsets.  
Need to find optimal partitioning.



OR



- What about this split?



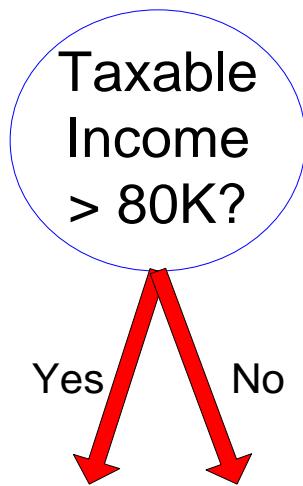
# Splitting Based on Continuous Attributes

---

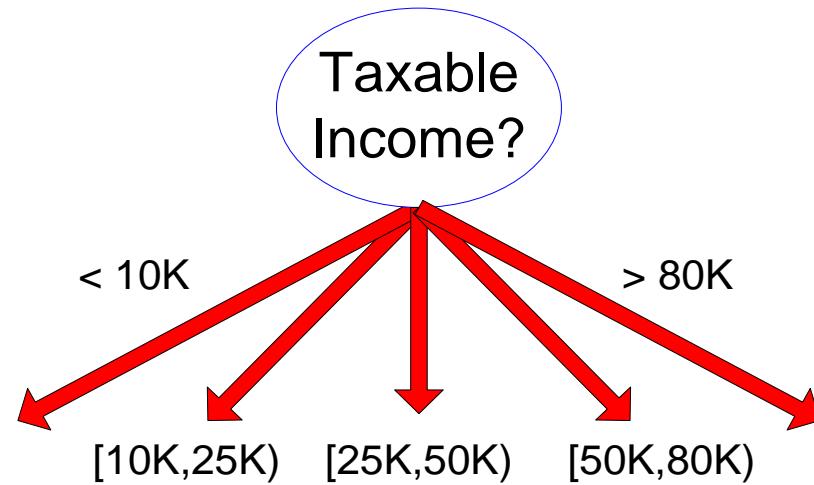
## □ Different ways of handling

- **Discretization** to form an ordinal categorical attribute
  - ◆ Static – discretize once at the beginning
  - ◆ Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
- **Binary Decision:**  $(A < v)$  or  $(A \geq v)$ 
  - ◆ consider all possible splits and finds the best cut
  - ◆ can be more compute intensive

# Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

# Tree Induction

---

---

## □ Greedy strategy.

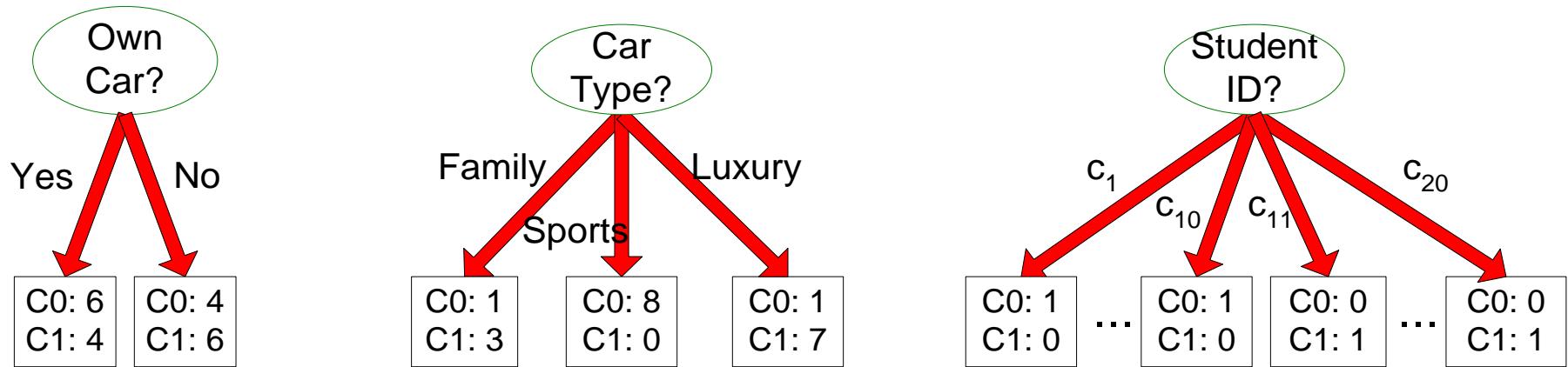
- Split the records based on an attribute test that optimizes certain criterion.

## □ Issues

- Determine how to split the records
  - ◆ How to specify the attribute test condition?
  - ◆ **How to determine the best split?**
- Determine when to stop splitting

# How to determine the Best Split

Before Splitting: 10 records of class 0,  
10 records of class 1



Which test condition is the best?

# How to determine the Best Split

---

- Greedy approach:
  - Nodes with **homogeneous** class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

**Non-homogeneous,**  
**High degree of impurity**

C0: 9
C1: 1

**Homogeneous,**  
**Low degree of impurity**

# Measures of Node Impurity

---

---

□ Gini Index

□ Entropy

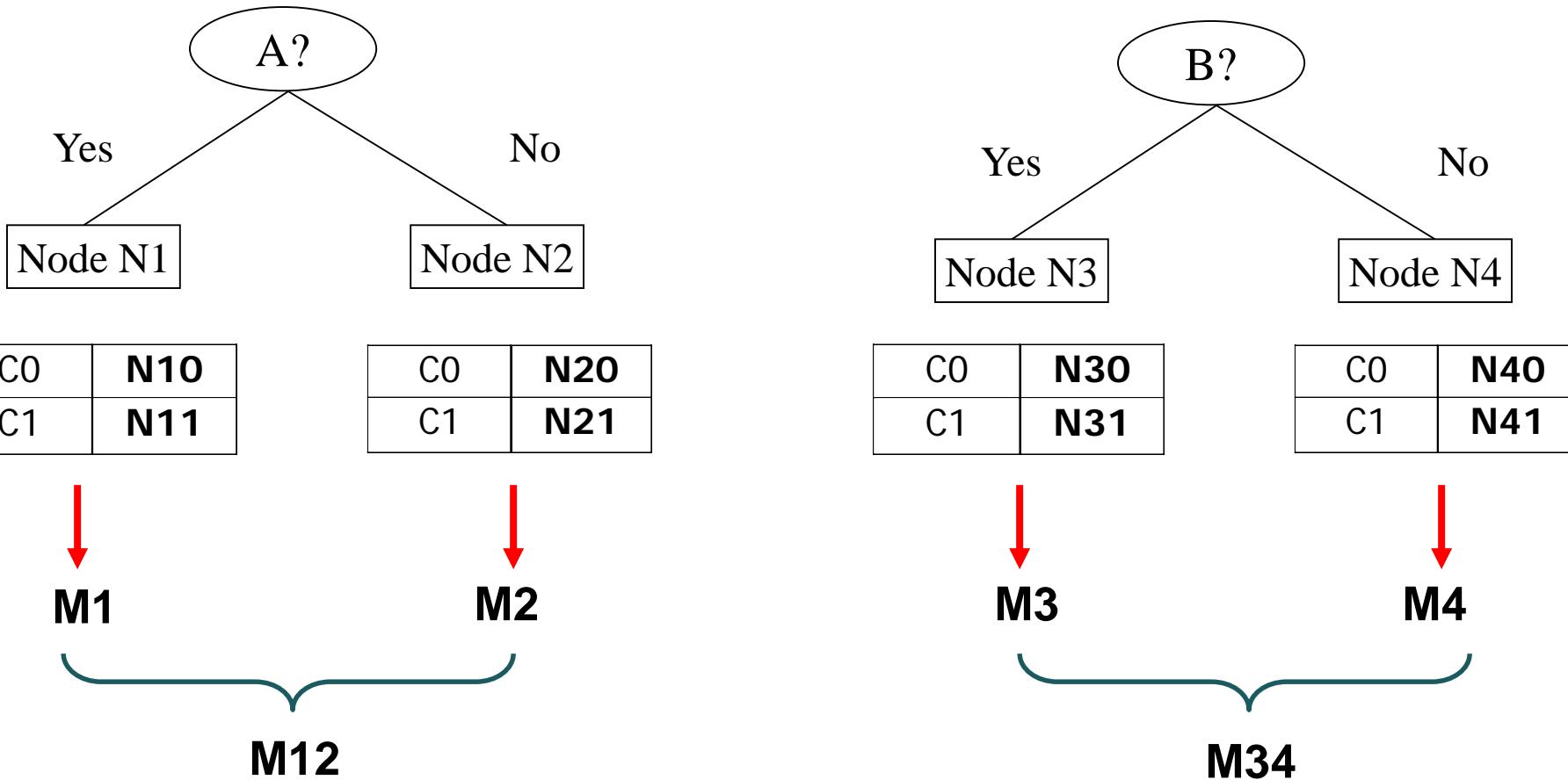
□ Misclassification error

# How to Find the Best Split

Before Splitting:

C0	N00
C1	N01

→ M0



$$\text{Gain} = M0 - M12 \text{ vs } M0 - M34$$

# Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

(NOTE:  $p(j | t)$  is the relative frequency of class j at node t).

- Maximum ( $1 - 1/n_c$ ) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
<b>Gini=0.000</b>	

C1	1
C2	5
<b>Gini=0.278</b>	

C1	2
C2	4
<b>Gini=0.444</b>	

C1	3
C2	3
<b>Gini=0.500</b>	

# Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

# Splitting Based on GINI

---

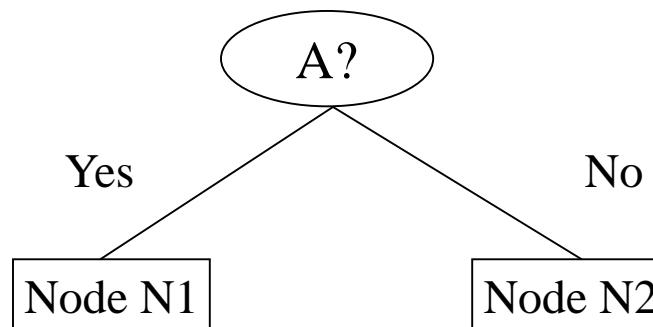
- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where,       $n_i$  = number of records at child i,  
                 $n$  = number of records at node p.

# Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



**Gini(N1)**

$$\begin{aligned} &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.408 \end{aligned}$$

**Gini(N2)**

$$\begin{aligned} &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.32 \end{aligned}$$

	N1	N2
C1	5	1
C2	2	4
<b>Gini=0.371</b>		

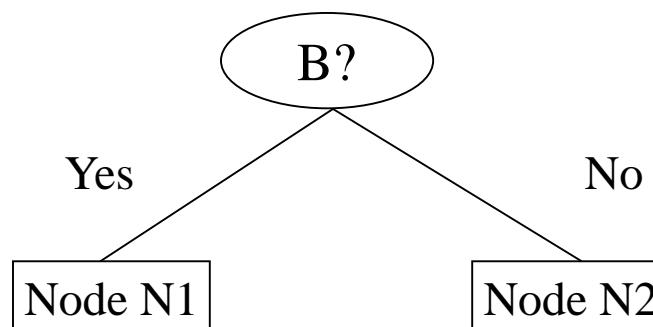
	Parent
C1	6
C2	6
<b>Gini = 0.500</b>	

**Gini(Children)**

$$\begin{aligned} &= 7/12 * 0.408 + \\ &\quad 5/12 * 0.32 \\ &= 0.371 \end{aligned}$$

# Binary Attributes: Computing GINI Index

- How about splitting based on B attribute?
- (Figure 4.14 page 161)



$$\begin{aligned} \text{Gini}(N1) &= 1 - (l)^2 - (l)^2 \\ &= \\ \text{Gini}(N2) &= 1 - (l)^2 - (l)^2 \\ &= \end{aligned}$$

	N1	N2
C1	3	4
C2	3	2
<b>Gini=?</b>		

	Parent
C1	6
C2	6
<b>Gini = 0.500</b>	

$$\begin{aligned} \text{Gini(Children)} &= \\ &= \end{aligned}$$

# Categorical Attributes: Computing Gini Index

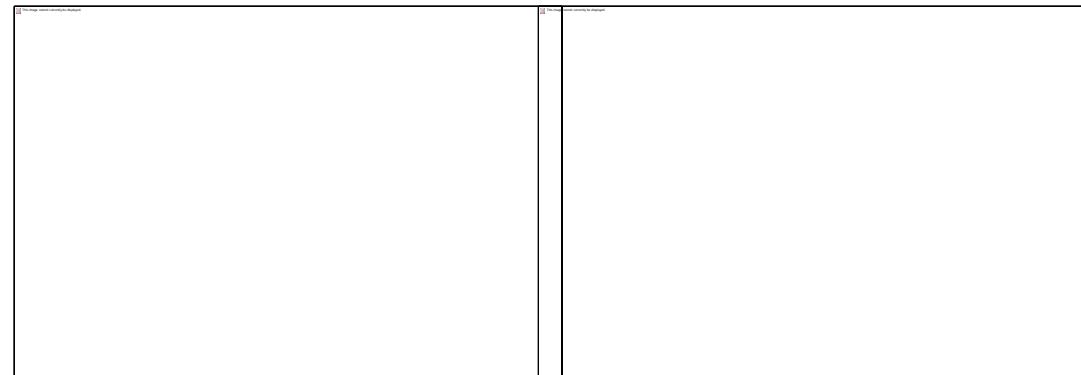
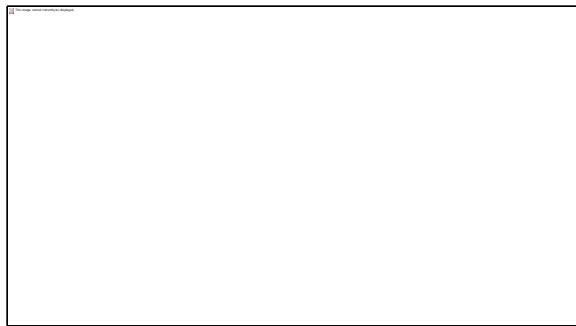
---

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split



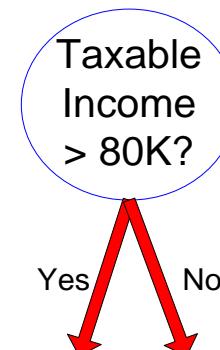
Two-way split  
(find best partition of values)



# Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A < v$  and  $A \geq v$
- Simple method to choose best  $v$ 
  - For each  $v$ , scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient!  
Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



# Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No		
Taxable Income												
Sorted Values	60	70	75	85	90	95	100	120	125	220		
Split Positions	55	65	72	80	87	92	97	110	122	172	230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	1	2	2	1	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420	

# Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

(NOTE:  $p(j | t)$  is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - ◆ Maximum ( $\log n_c$ ) when records are equally distributed among all classes implying least information
  - ◆ Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

# Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Entropy} = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

# Splitting Based on INFO...

## □ Information Gain:

$$GAIN_{split} = Entropy(p) - \left( \sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

$n_i$  is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

# Splitting Based on INFO...

## □ Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

$n_i$  is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

# Splitting Criteria based on Classification Error

---

- Classification error at a node  $t$  :

$$Error(t) = 1 - \max_i P(i | t)$$

- Measures misclassification error made by a node.
  - ◆ Maximum ( $1 - 1/n_c$ ) when records are equally distributed among all classes, implying least interesting information
  - ◆ Minimum (0.0) when all records belong to one class, implying most interesting information

# Examples for Computing Error

---

---

$$Error(t) = 1 - \max_i P(i | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

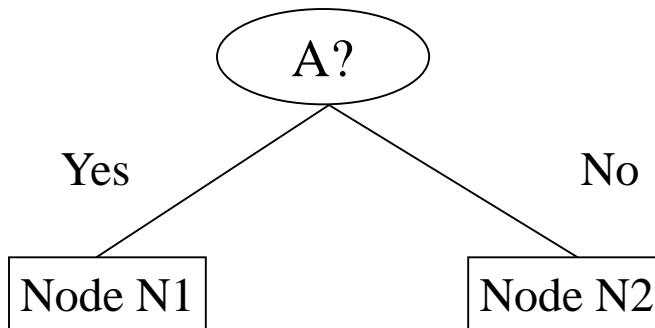
$$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Error} = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

# Gini



	<b>Parent</b>
C1	<b>7</b>
C2	<b>3</b>
<b>Gini = 0.42</b>	

**Gini(N1)**

$$= 1 - (3/3)^2 - (0/3)^2$$

$$= 0$$

**Gini(N2)**

$$= 1 - (4/7)^2 - (3/7)^2$$

$$= 0.489$$

	<b>N1</b>	<b>N2</b>
C1	<b>3</b>	<b>4</b>
C2	<b>0</b>	<b>3</b>
<b>Gini=0.342</b>		

**Gini(Children)**

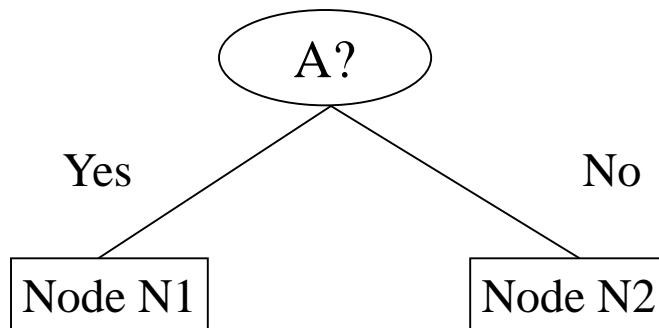
$$= 3/10 * 0$$

$$+ 7/10 * 0.489$$

$$= 0.342$$

**Gini improves !!**

# Misclassification Error



	<b>Parent</b>
C1	<b>7</b>
C2	<b>3</b>
<b>M.Error = 0.3</b>	

$$M.E(N_1)$$

$$= 0$$

$$M.E(N_2)$$

$$= 1/5$$

$$= 0.2$$

	N1	N2
C1	5	4
C2	0	1
<b>M.E=0.1</b>		

$$M.E(\text{Children})$$

$$= 5/10 * 0$$

$$+ 5/10 * 0.2$$

$$= 0.1$$

# Tree Induction

---

---

## □ Greedy strategy.

- Split the records based on an attribute test that optimizes certain criterion.

## □ Issues

- Determine how to split the records
  - ◆ How to specify the attribute test condition?
  - ◆ How to determine the best split?
- Determine when to stop splitting

# Stopping Criteria for Tree Induction

---

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

# Decision Tree Based Classification

---

## □ Advantages:

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

# Example: C4.5

---

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.
- You can download the software from:  
<http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz>

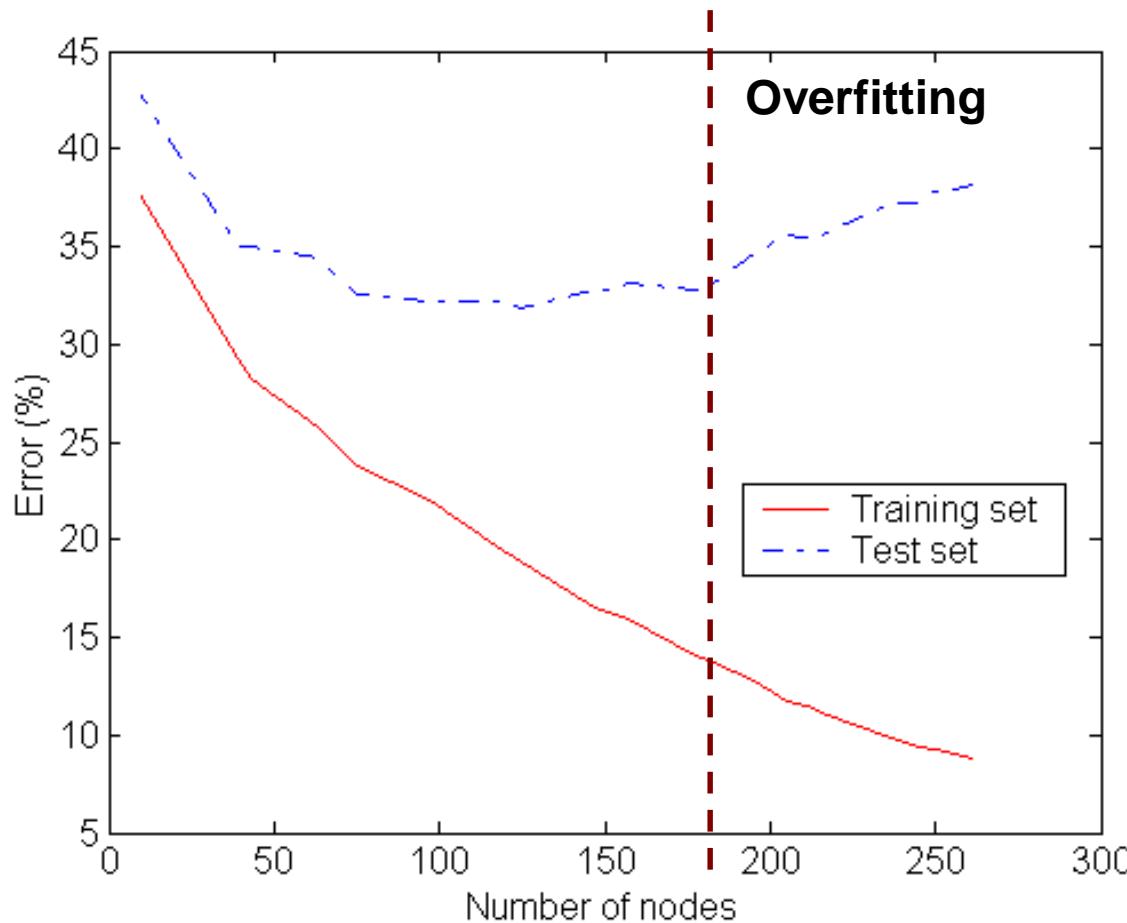
# Practical Issues of Classification

---

---

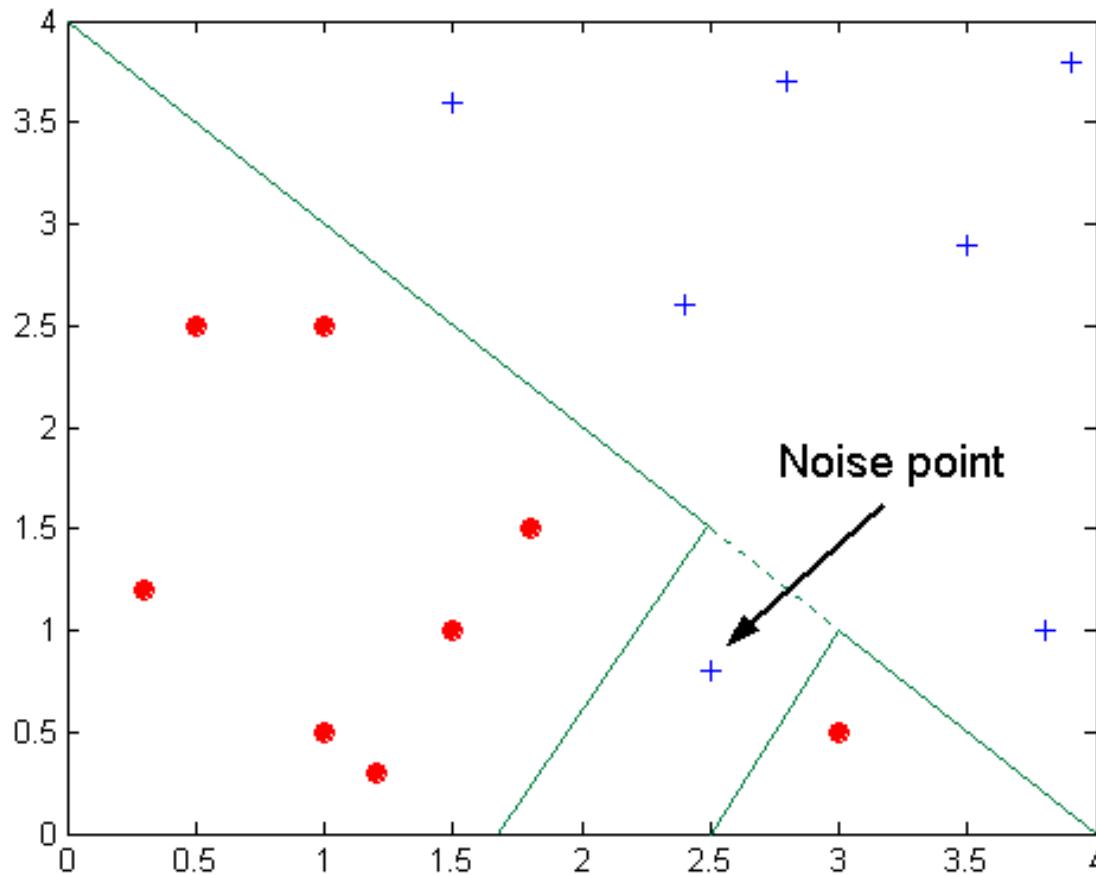
- Underfitting and Overfitting
- Missing Values
- Costs of Classification

# Underfitting and Overfitting



**Underfitting:** when model is too simple, both training and test errors are large

# Overfitting due to Noise



Decision boundary is distorted by noise point

# Overfitting due to Noise

**Table 4.3.** An example training set for classifying mammals. Class labels with asterisk symbols represent mislabeled records.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

\* Bats and Whales are misclassified; non-mammals instead of mammals.

# Overfitting due to Noise

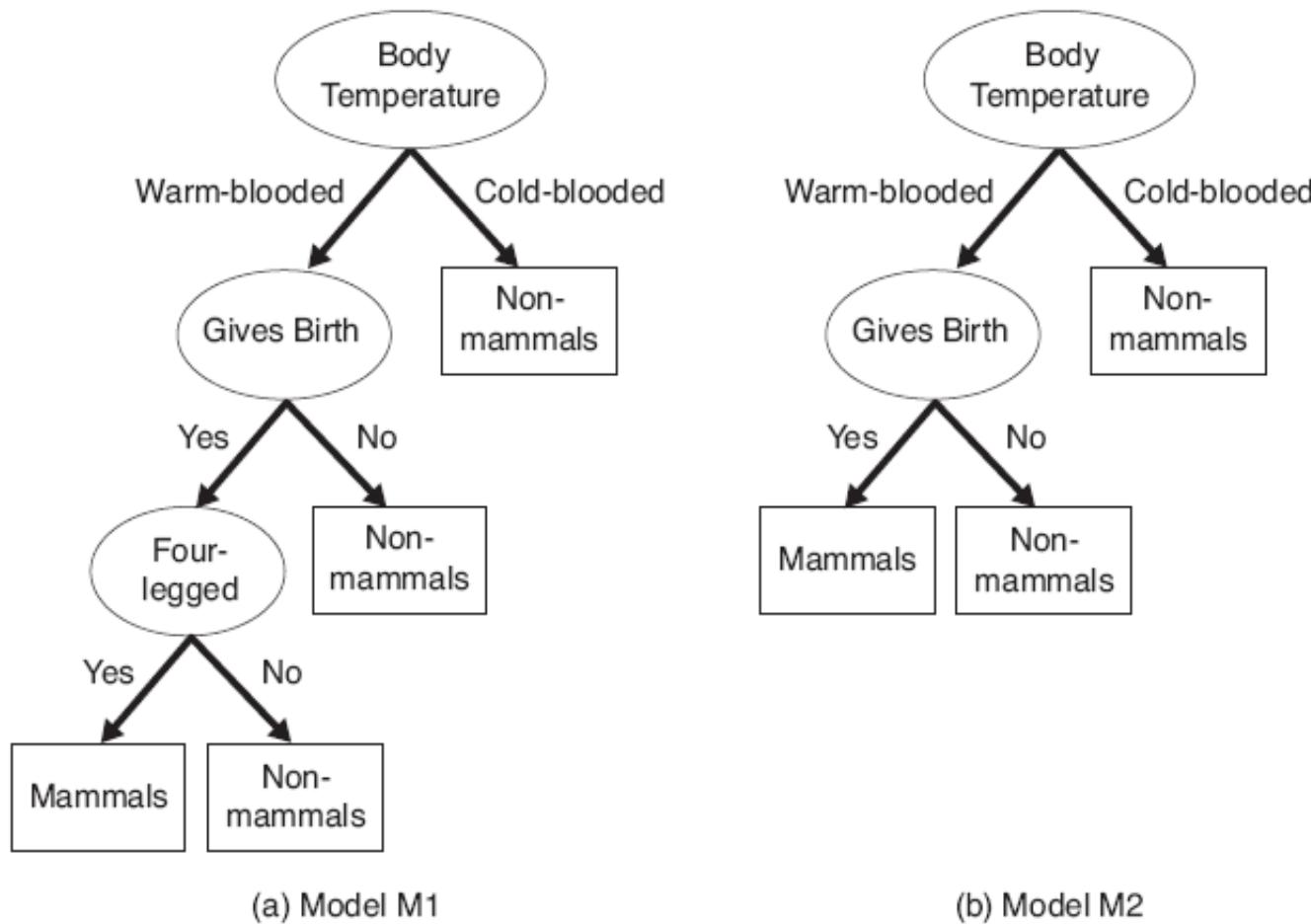


Figure 4.25. Decision tree induced from the data set shown in Table 4.3.

# Overfitting due to Noise

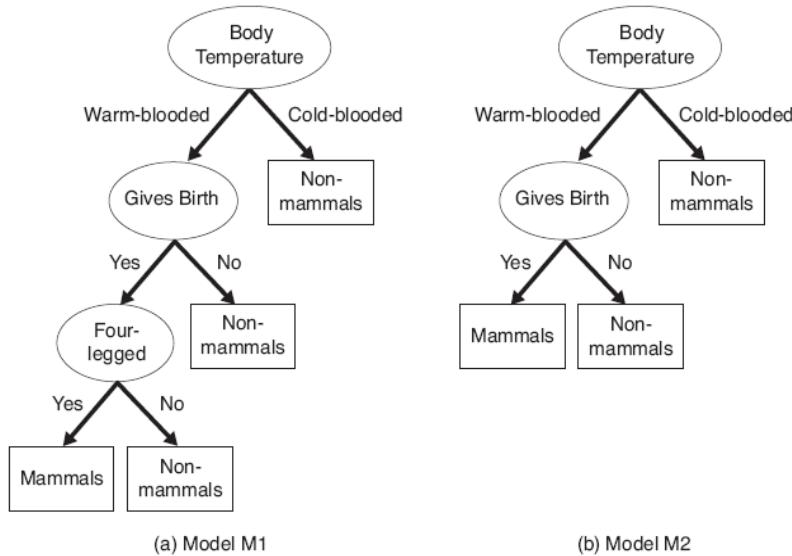
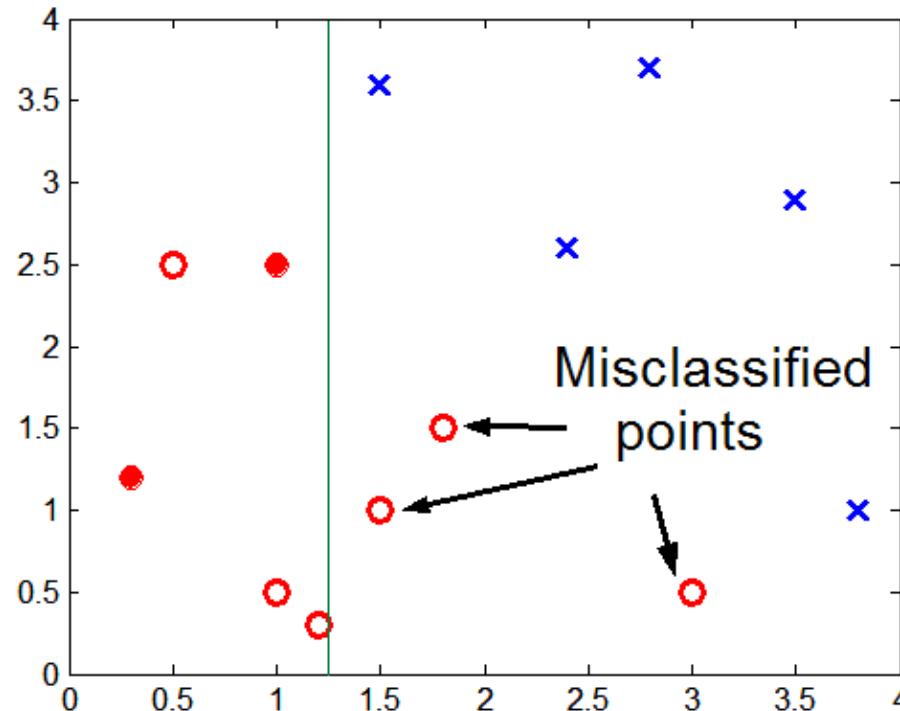


Figure 4.25. Decision tree induced from the data set shown in Table 4.3.

- Decision tree perfectly fits training data (training error=0)
- But error rate on test data is 30%.

- Both humans and dolphins were misclassified as non-mammals b/c Body Temp, Gives\_Birth and Four-legged values are identical to mislabeled records in training set.
- Spiny anteaters represent an exceptional case (every warm-blooded with no gives\_birth is non-mammal in TR\_Set)

# Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

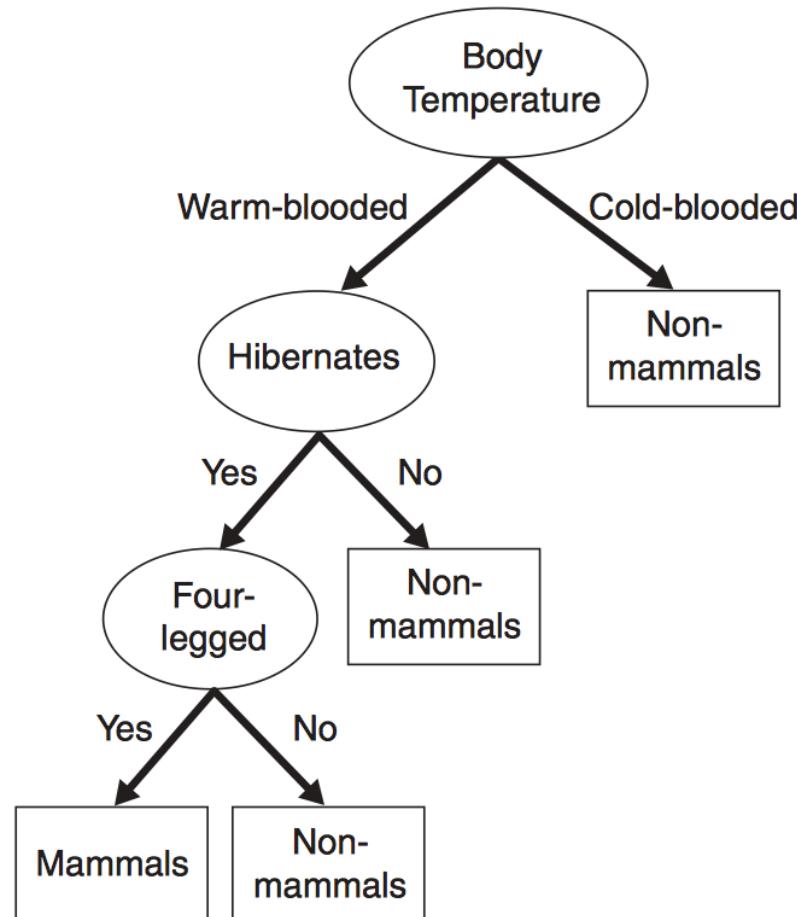
# Overfitting due to Insufficient Examples

**Table 4.5.** An example training set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
salamander	cold-blooded	no	yes	yes	no
guppy	cold-blooded	yes	no	no	no
eagle	warm-blooded	no	no	no	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes

Models that make their classification decisions based on a small number of training records are also susceptible to overfitting.

# Overfitting due to Insufficient Examples



**Figure 4.26.** Decision tree induced from the data set shown in Table 4.5.

# Overfitting due to Insufficient Examples

---

- All of these training records are labeled correctly and its training error is zero, its error rate on the test set is 30%.
- Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded vertebrates that do not hibernate as non-mammals.
- The tree arrives at this classification decision because there is only one training record, which is an eagle, with such characteristics.
- This example clearly demonstrates the danger of making wrong predictions when there are not enough representative examples at the leaf nodes of a decision tree.

# Notes on Overfitting

---

---

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

# Estimating Generalization Errors

---

- Re-substitution errors: error on training ( $\sum e(t)$ )
- Generalization errors: error on testing ( $\sum e'(t)$ )
- Methods for estimating generalization errors:
  - Optimistic approach:  $e'(t) = e(t)$
  - Pessimistic approach:
    - ◆ For each leaf node:  $e'(t) = (e(t)+0.5)$
    - ◆ Total errors:  $e'(T) = e(T) + N \times 0.5$  (N: number of leaf nodes)
    - ◆ For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):  
Training error =  $10/1000 = 1\%$   
Generalization error =  $(10 + 30 \times 0.5)/1000 = 2.5\%$
  - Reduced error pruning (REP):
    - ◆ uses validation data set to estimate generalization error

# Occam's Razor

---

---

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
  
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
  
- Therefore, one should include model complexity when evaluating a model

# How to Address Overfitting

---

## □ Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - ◆ Stop if all instances belong to the same class
  - ◆ Stop if all the attribute values are the same
- More restrictive conditions:
  - ◆ Stop if number of instances is less than some user-specified threshold
  - ◆ Stop if class distribution of instances are independent of the available features (e.g., using  $\chi^2$  test)
  - ◆ Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

# How to Address Overfitting...

---

## □ Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the sub-tree

# Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

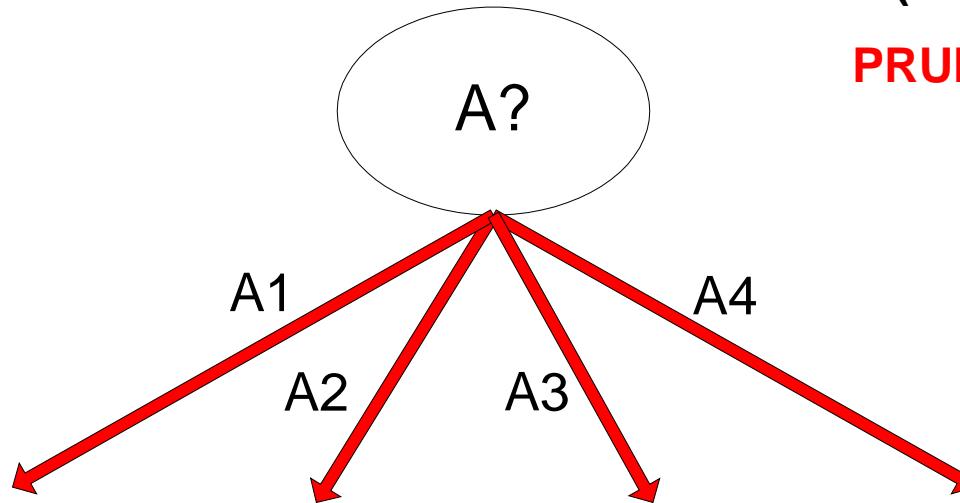
Pessimistic error =  $(10 + 0.5)/30 = 10.5/30$

Training Error (After splitting) = 7/30

Pessimistic error (After splitting)

$$= (7 + 4 \times 0.5)/30 = 9/30$$

**PRUNE OR DO NOT PRUNE**



Class = Yes	8
Class = No	4

Class = Yes	2
Class = No	5

Class = Yes	6
Class = No	1

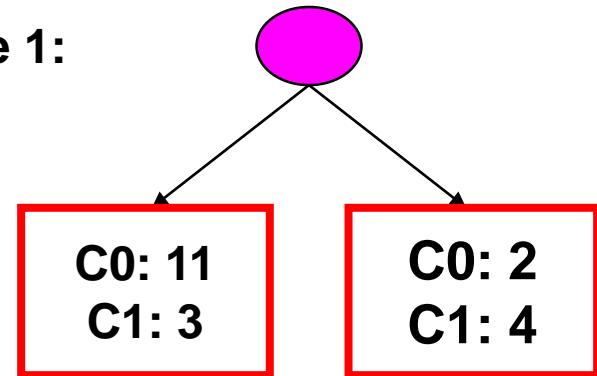
Class = Yes	4
Class = No	0

# Examples of Post-pruning

- Optimistic error?

Don't prune for both cases

**Case 1:**



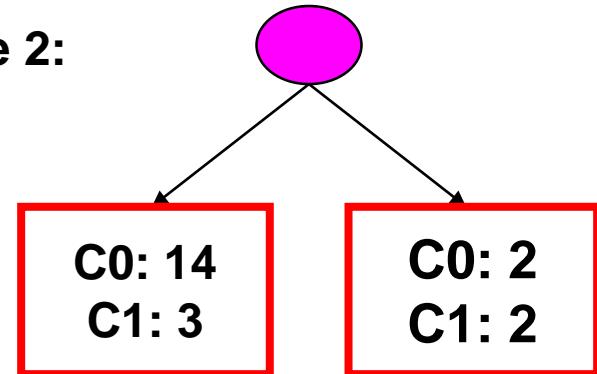
- Pessimistic error?

Don't prune case 1, prune case 2

- Reduced error pruning?

Depends on validation set

**Case 2:**



# Handling Missing Attribute Values

---

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

# Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)

$$= -0.3 \log(0.3) - (0.7)\log(0.7) = 0.8813$$

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)

$$= -(2/6)\log(2/6) - (4/6)\log(4/6) = 0.9183$$

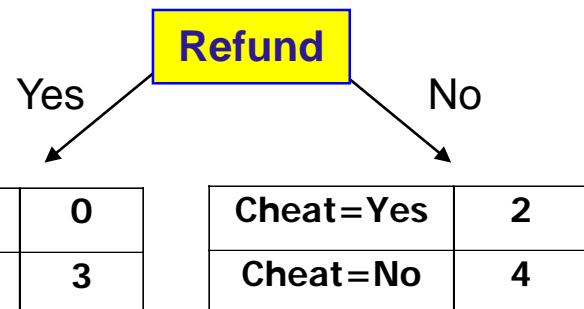
Entropy(Children)

$$= 0.3 (0) + 0.6 (0.9183) = 0.551$$

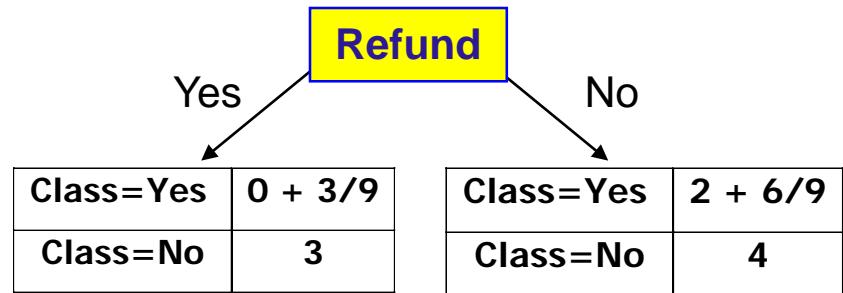
$$\text{Gain} = 0.9 \times (0.8813 - 0.551) = 0.3303$$

# Distribute Instances

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No



Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

# A Famous Statistics Problem

---

---

## □ Monty Hall Problem

# Exam I Review

---

---

- What is Data Mining?
- Application of Data Mining in Various Domains and Cases?
- Types of Data?
- Why to know your data is important?
- Sampling, Aggregation,...
- Curse of Dimensionality
- Similarity calculations (expect 1 question)
- Interpreting visualizations
- Basics of PCA

# Exam I Review

---

---

- Classification: Accuracy vs Error Rate
- Entropy, Gini, Error & Gain calculations
- Overfitting concepts: what, why, how
- Generalization errors (pessimistic/optimistic)

# Data Fragmentation

---

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

# Other Issues

---

---

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

# Search Strategy

---

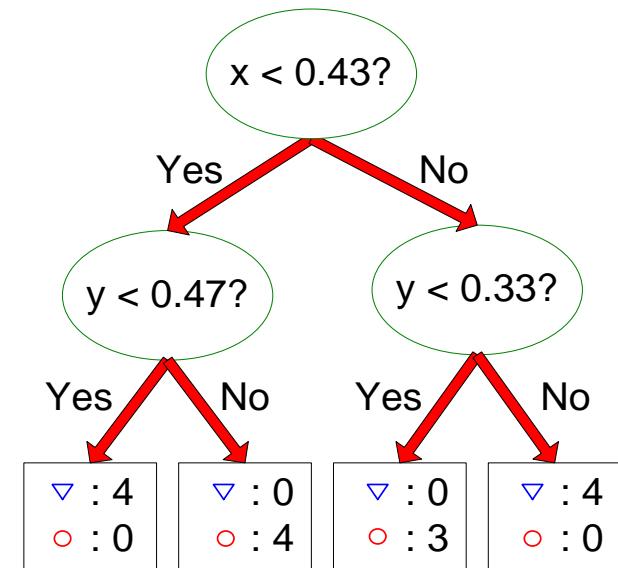
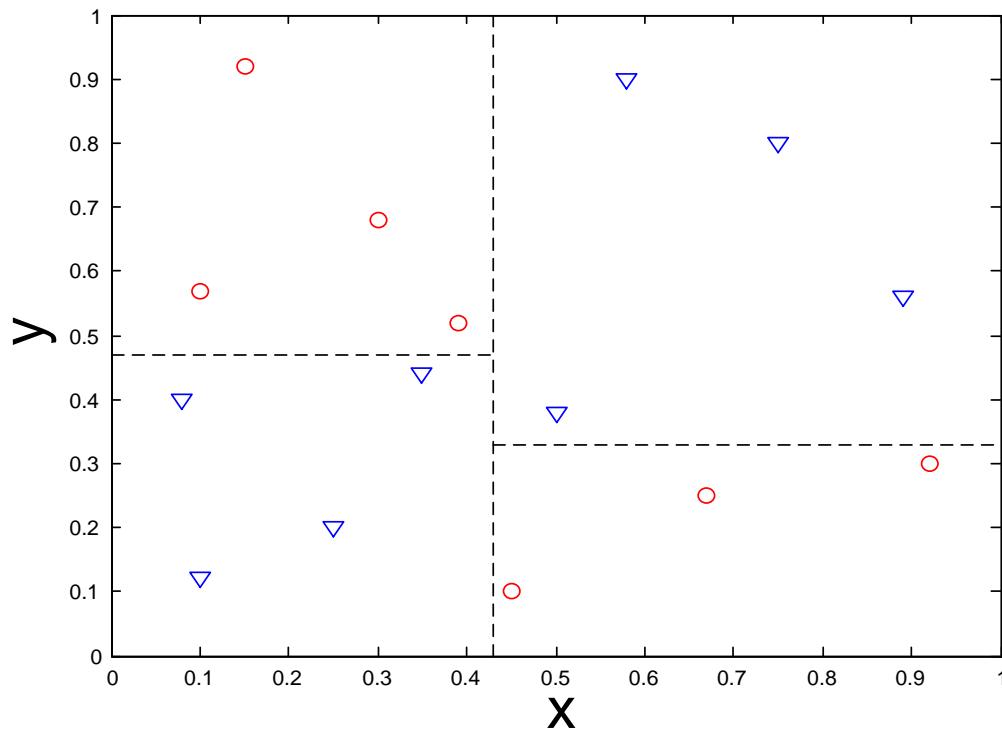
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

# Expressiveness

---

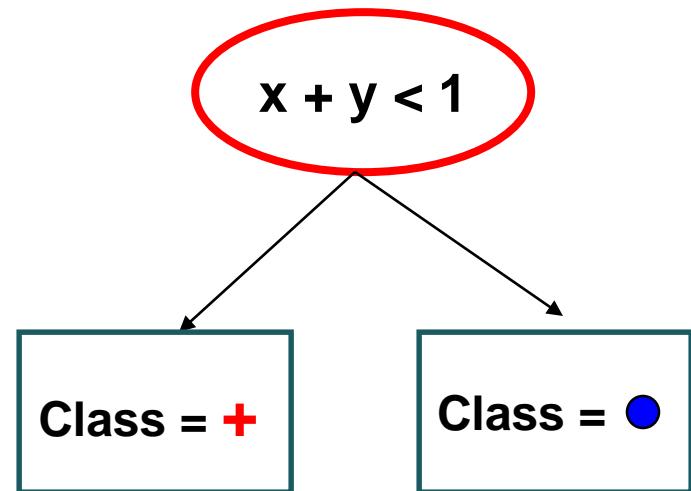
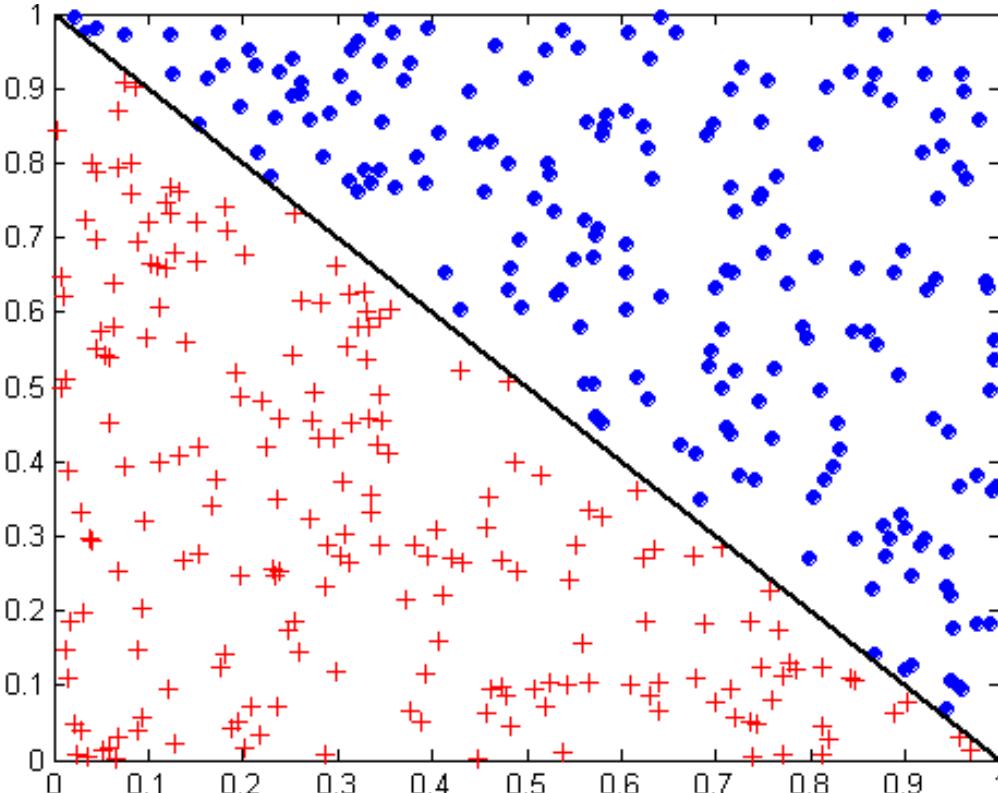
- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - ◆ Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - ◆ For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

# Decision Boundary



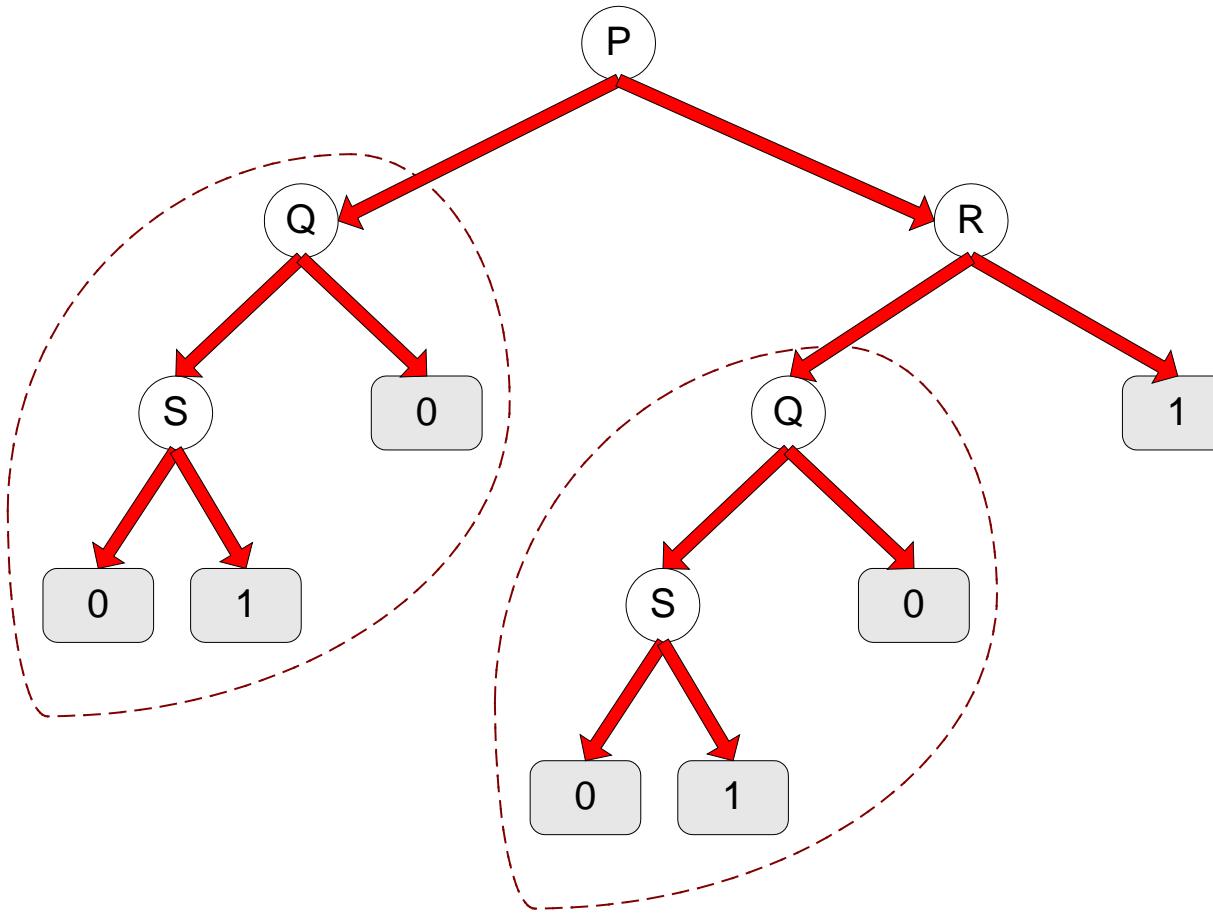
- Border line between two neighboring regions of different classes is known as **decision boundary**
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

# Oblique Decision Trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

# Tree Replication



- Same subtree appears in multiple branches

# Model Evaluation

---

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# Model Evaluation

---

## □ Metrics for Performance Evaluation

- How to evaluate the performance of a model?

## □ Methods for Performance Evaluation

- How to obtain reliable estimates?

## □ Methods for Model Comparison

- How to compare the relative performance among competing models?

# Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a	b
	Class>No	c	d

a: TP (true positive)  
b: FN (false negative)  
c: FP (false positive)  
d: TN (true negative)

# Metrics for Performance Evaluation...

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

# Limitation of Accuracy

---

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
  
- If model predicts everything to be class 0, accuracy is  $9990/10000 = 99.9\%$ 
  - Accuracy is misleading because model does not detect any class 1 example

# Cost Matrix

		PREDICTED CLASS		
		C(i j)	Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	C(Yes Yes)	C(No Yes)	
	Class>No	C(Yes No)	C(No No)	

$C(i|j)$ : Cost of misclassifying class  $j$  example as class  $i$

# Computing Cost of Classification

Cost Matrix		PREDICTED CLASS	
ACTUAL CLASS	C(i  j)	+	-
	+	-1	100
	-	1	0

Model M <sub>1</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Model M <sub>2</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

# Cost vs Accuracy

Count	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class>No
	Class=Yes	a
	Class>No	d

Accuracy is proportional to cost if

1.  $C(Yes|No)=C(No|Yes) = q$
2.  $C(Yes|Yes)=C(No|No) = p$

$$N = a + b + c + d$$

$$\text{Accuracy} = (a + d)/N$$

Cost	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class>No
	Class=Yes	p
	Class>No	q

$$\text{Cost} = p (a + d) + q (b + c)$$

$$= p (a + d) + q (N - a - d)$$

$$= q N - (q - p)(a + d)$$

$$= N [q - (q-p) \times \text{Accuracy}]$$

# Model Evaluation

---

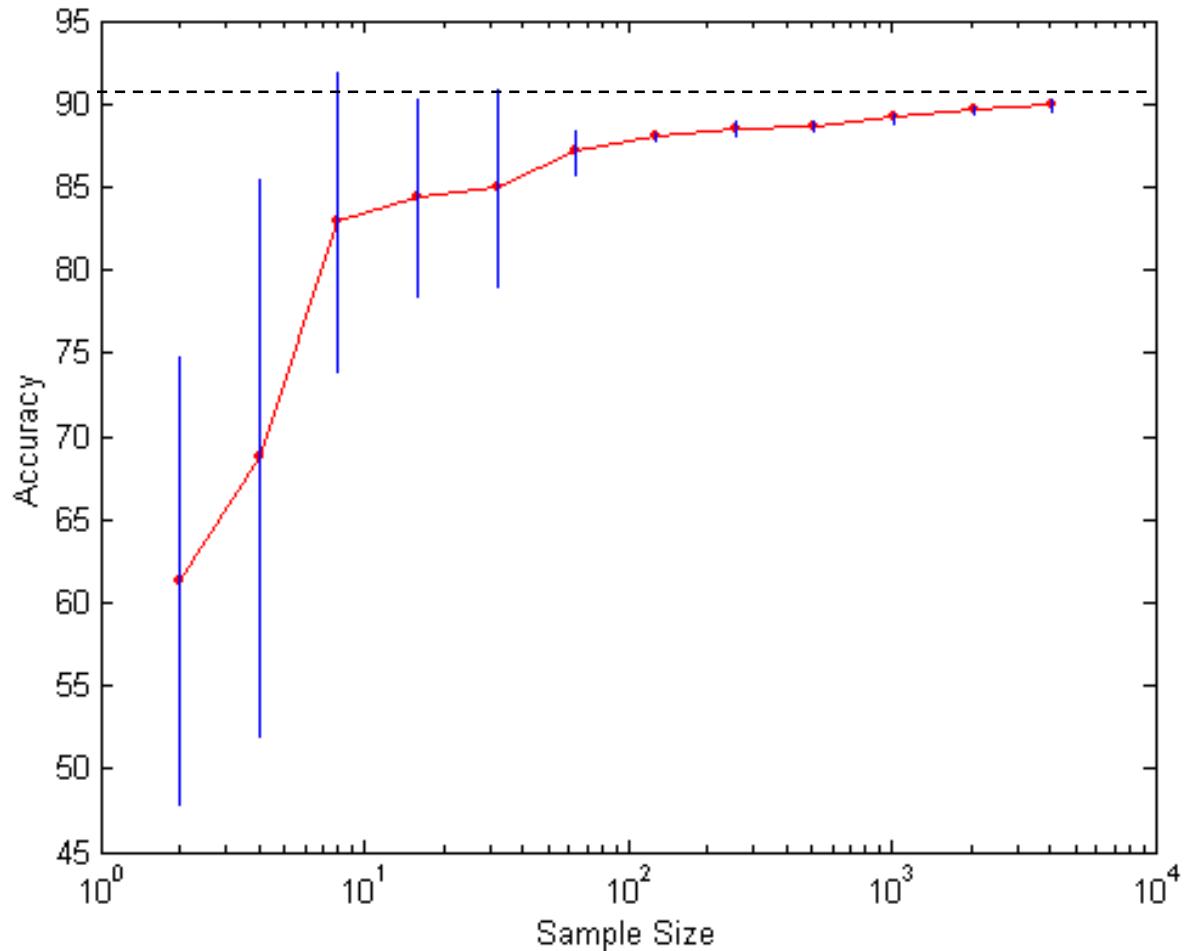
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# Methods for Performance Evaluation

---

- How to obtain a reliable estimate of performance?
  
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

# Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)
- Effect of small sample size:
  - Bias in the estimate
  - Variance of estimate

# Methods of Estimation

---

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into  $k$  disjoint subsets
  - $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - Leave-one-out:  $k=n$
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

# Model Evaluation

---

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

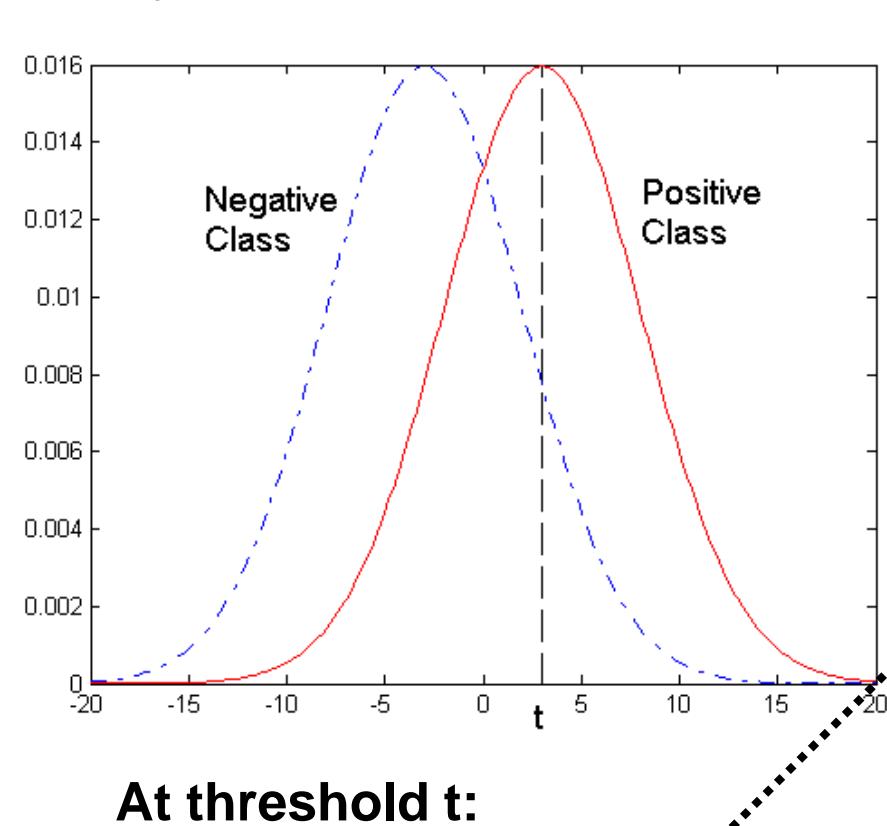
# ROC (Receiver Operating Characteristic)

---

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

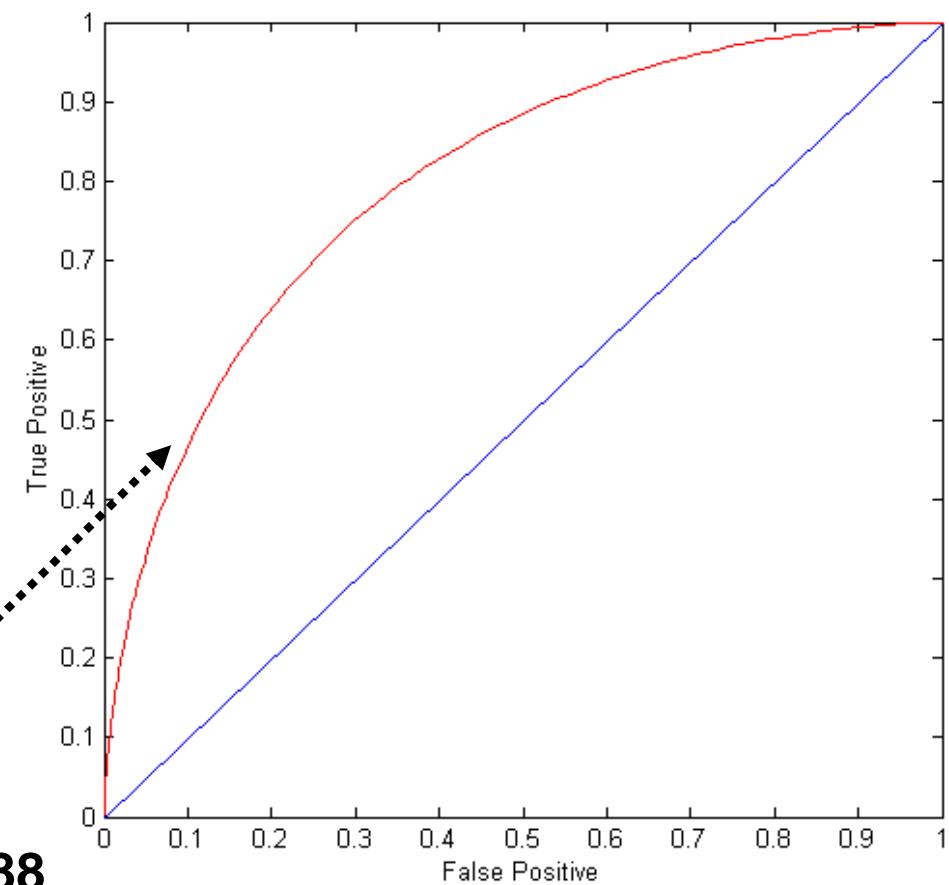
# ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at  $x > t$  is classified as positive



At threshold  $t$ :

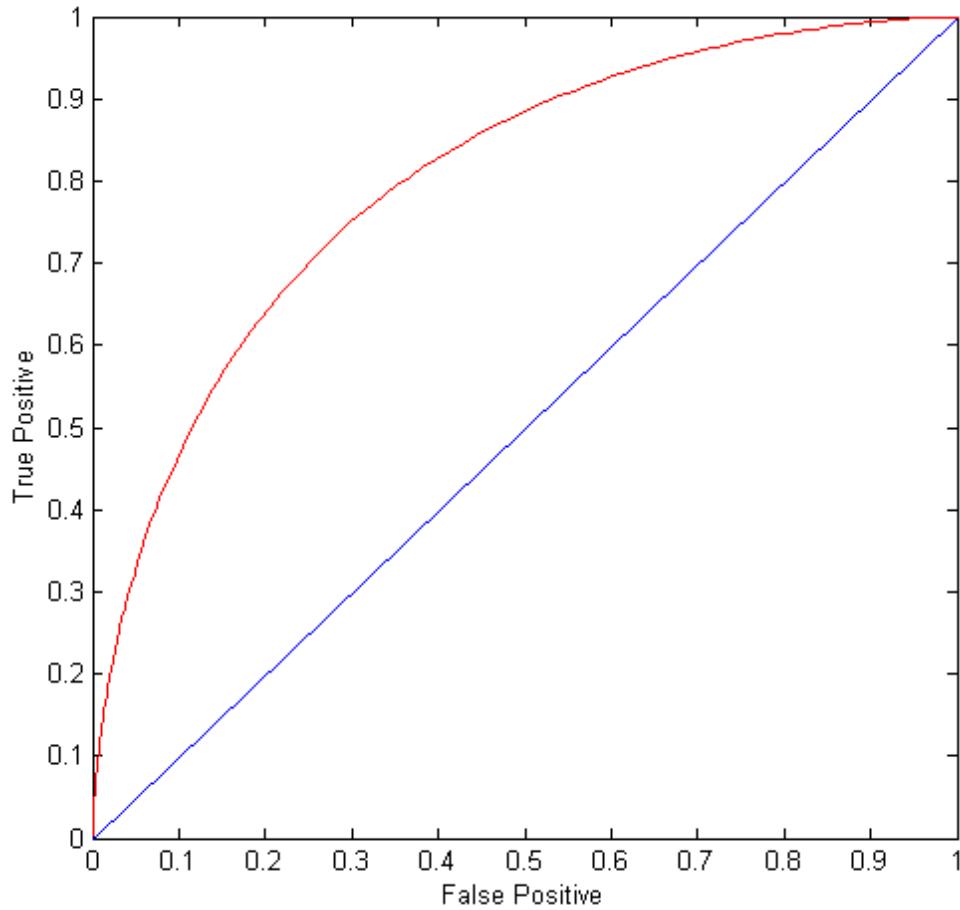
$\text{TP}=0.5, \text{FN}=0.5, \text{FP}=0.12, \text{FN}=0.88$



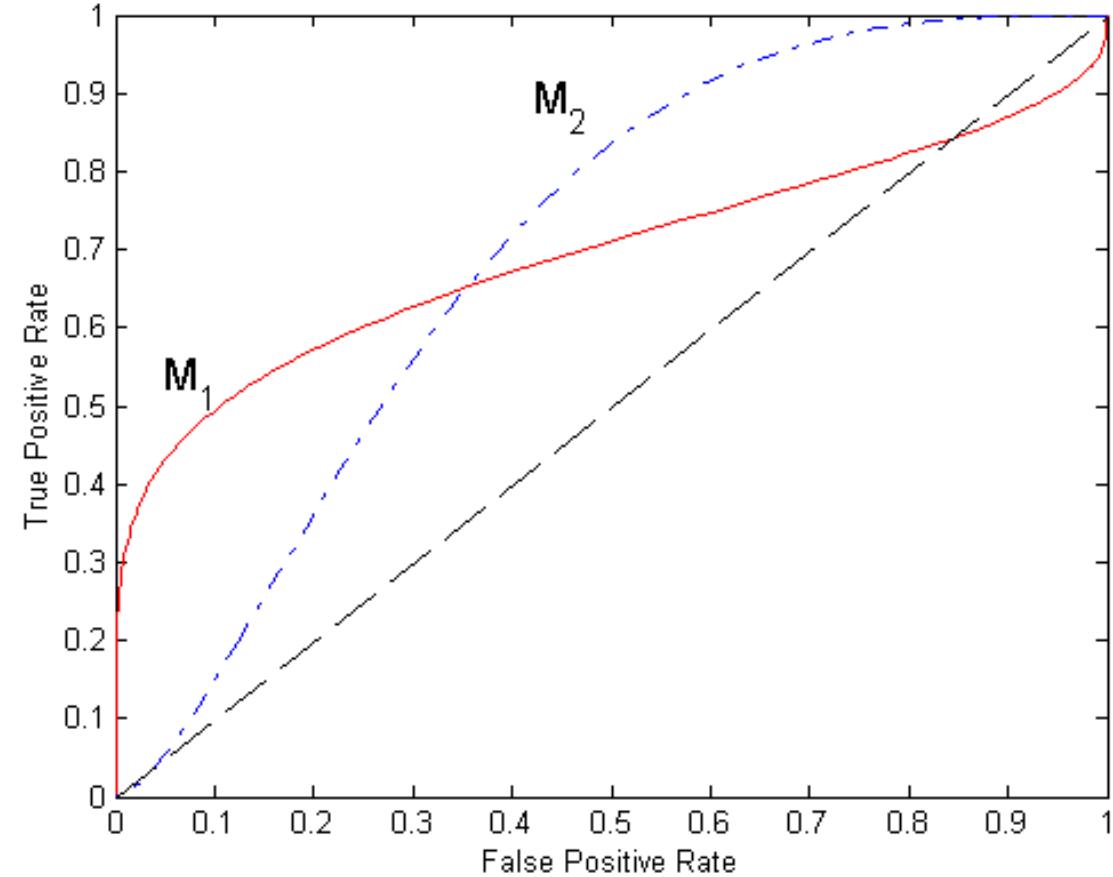
# ROC Curve

(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
  
- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - ◆ prediction is opposite of the true class



# Using ROC for Model Comparison



- No model consistently outperform the other
  - $M_1$  is better for small FPR
  - $M_2$  is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

# How to Construct an ROC curve

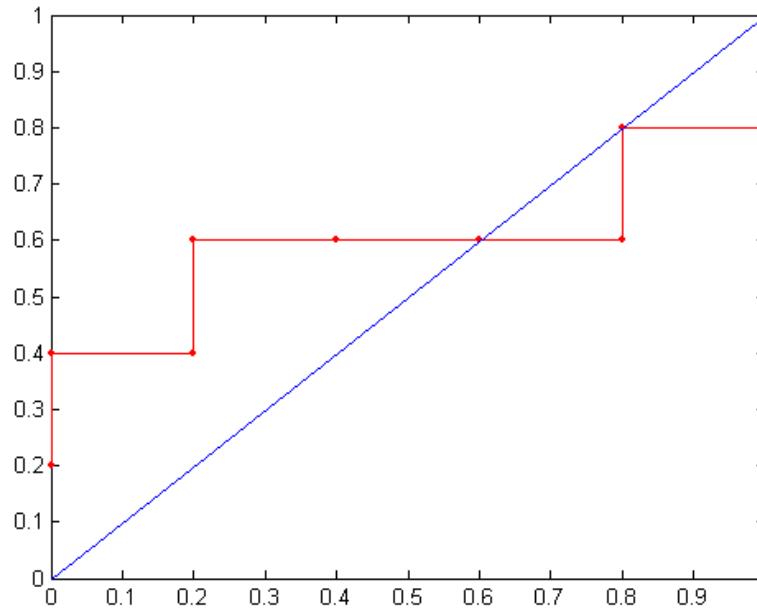
Instance	$P(+ A)$	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance  $P(+|A)$
- Sort the instances according to  $P(+|A)$  in decreasing order
- Apply threshold at each unique value of  $P(+|A)$
- Count the number of TP, FP, TN, FN at each threshold
- TP rate,  $TPR = TP/(TP+FN)$
- FP rate,  $FPR = FP/(FP + TN)$

# How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00	
TP	5	4	4	3	3	3	3	2	2	1	0	0
FP	5	5	4	4	3	2	1	1	0	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5	5
→ TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0	
→ FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0	

ROC Curve:



# Test of Significance

---

---

## □ Given two models:

- Model M1: accuracy = 85%, tested on 30 instances
- Model M2: accuracy = 75%, tested on 5000 instances

## □ Can we say M1 is better than M2?

- How much confidence can we place on accuracy of M1 and M2?
- Can the difference in performance measure be explained as a result of random fluctuations in the test set?

# Confidence Interval for Accuracy

---

- Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - ◆  $x \sim \text{Bin}(N, p)$       $x$ : number of correct predictions
    - ◆ e.g: Toss a fair coin 50 times, how many heads would turn up?  
Expected number of heads =  $N \times p = 50 \times 0.5 = 25$
- Given  $x$  (# of correct predictions) or equivalently,  $\text{acc} = x/N$ , and  $N$  (# of test instances),

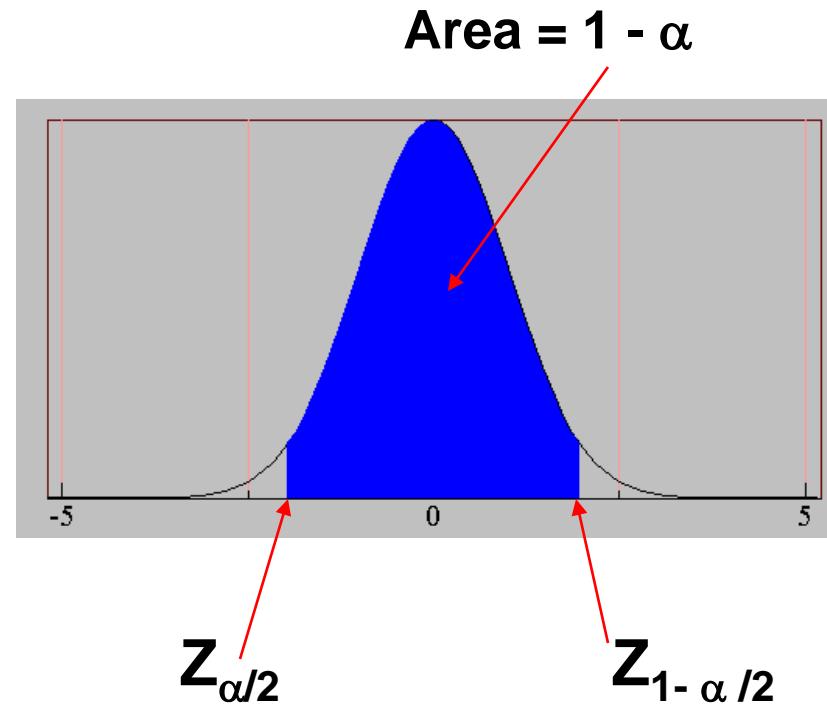
Can we predict  $p$  (true accuracy of model)?

# Confidence Interval for Accuracy

- For large test sets ( $N > 30$ ),

- acc has a normal distribution with mean  $p$  and variance  $p(1-p)/N$

$$P(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}) = 1 - \alpha$$



- Confidence Interval for  $p$ :

$$p = \frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm \sqrt{Z_{\alpha/2}^2 + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z_{\alpha/2}^2)}$$

# Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:

- $N=100$ , acc = 0.8
- Let  $1-\alpha = 0.95$  (95% confidence)
- From probability table,  $Z_{\alpha/2}=1.96$

N	50	100	500	1000	5000
p(lower)	0.670	0.711	0.763	0.774	0.789
p(upper)	0.888	0.866	0.833	0.824	0.811

1- $\alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

# Comparing Performance of 2 Models

---

- Given two models, say M1 and M2, which is better?
  - M1 is tested on D1 (size=n1), found error rate =  $e_1$
  - M2 is tested on D2 (size=n2), found error rate =  $e_2$
  - Assume D1 and D2 are independent
  - If n1 and n2 are sufficiently large, then
$$e_1 \sim N(\mu_1, \sigma_1)$$
$$e_2 \sim N(\mu_2, \sigma_2)$$
  - Approximate:  $\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$

# Comparing Performance of 2 Models

---

- To test if performance difference is statistically significant:  $d = e_1 - e_2$

- $d \sim N(d_t, \sigma_t)$  where  $d_t$  is the true difference
  - Since  $D_1$  and  $D_2$  are independent, their variance adds up:

$$\begin{aligned}\sigma_t^2 &= \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2 \\ &= \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}\end{aligned}$$

- At  $(1-\alpha)$  confidence level,  $d_t = d \pm Z_{\alpha/2} \hat{\sigma}_t$

# An Illustrative Example

---

- Given: M1:  $n_1 = 30, e_1 = 0.15$   
M2:  $n_2 = 5000, e_2 = 0.25$
- $d = |e_2 - e_1| = 0.1$  (2-sided test)

$$\hat{\sigma}_d = \sqrt{\frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000}} = 0.0043$$

- At 95% confidence level,  $Z_{\alpha/2}=1.96$

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

=> Interval contains 0 => difference may not be statistically significant

# An Illustrative Example

---

---

The following table shows the values of  $Z_{\alpha/2}$  at different confidence levels:

$1 - \alpha$	0.99	0.98	0.95	0.9	0.8	0.7	0.5
$Z_{\alpha/2}$	2.58	2.33	1.96	1.65	1.28	1.04	0.67

# Comparing Performance of 2 Algorithms

---

- Each learning algorithm may produce k models:
  - L1 may produce M<sub>11</sub> , M<sub>12</sub>, ..., M<sub>1k</sub>
  - L2 may produce M<sub>21</sub> , M<sub>22</sub>, ..., M<sub>2k</sub>

- If models are generated on the same test sets D<sub>1</sub>,D<sub>2</sub>, ..., D<sub>k</sub> (e.g., via cross-validation)

- For each set: compute  $d_j = e_{1j} - e_{2j}$
- $d_j$  has mean  $d_t$  and variance  $\sigma_t$
- Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^k (d_j - \bar{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha, k-1} \hat{\sigma}_t$$

# Data Mining Classification: Alternative Techniques

---

Lecture Notes for Chapter 5

Introduction to Data Mining

by

Tan, Steinbach, Kumar

# Rule-Based Classifier

---

- Classify records by using a collection of “if... then...” rules
- Rule:  $(\textit{Condition}) \rightarrow y$ 
  - where
    - ◆ *Condition* is a conjunctions of attributes
    - ◆  $y$  is the class label
  - LHS: rule antecedent or condition
  - RHS: rule consequent
  - Examples of classification rules:
    - ◆  $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
    - ◆  $(\text{Taxable Income} < 50K) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Evade}=\text{No}$

# Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

# Application of Rule-Based Classifier

---

- A rule  $r$  **covers** an instance  $\mathbf{x}$  if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

# Rule Coverage and Accuracy

- Coverage of a rule:
  - Fraction of records that satisfy the antecedent of a rule
- Accuracy of a rule:
  - Fraction of records that satisfy both the antecedent and consequent of a rule

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$(\text{Status}=\text{Single}) \rightarrow \text{No}$

**Coverage = 40%, Accuracy = 50%**

# How does Rule-based Classifier Work?

---

R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds

R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes

R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals

R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles

R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

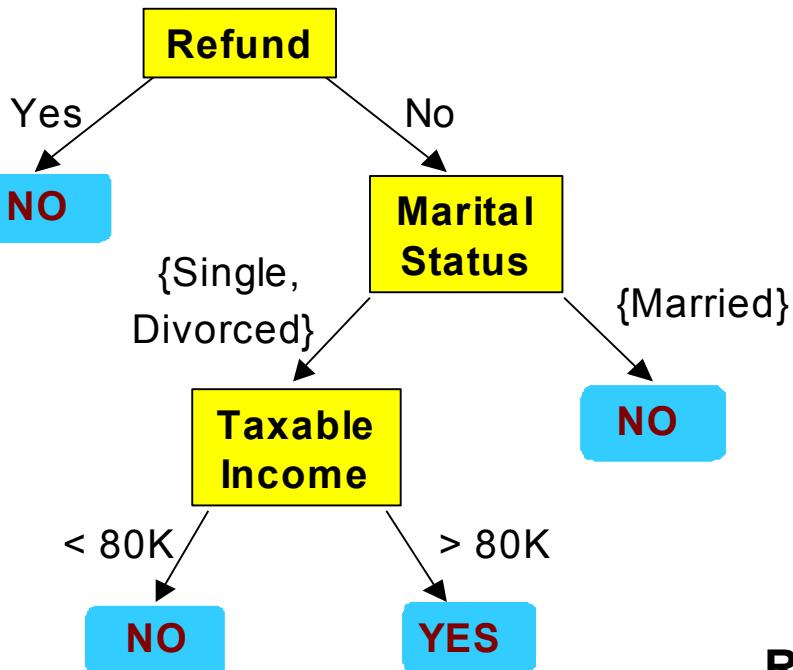
A dogfish shark triggers none of the rules

# Characteristics of Rule-Based Classifier

---

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule
  
- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

# From Decision Trees To Rules



## Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced},  
Taxable Income<80K) ==> No

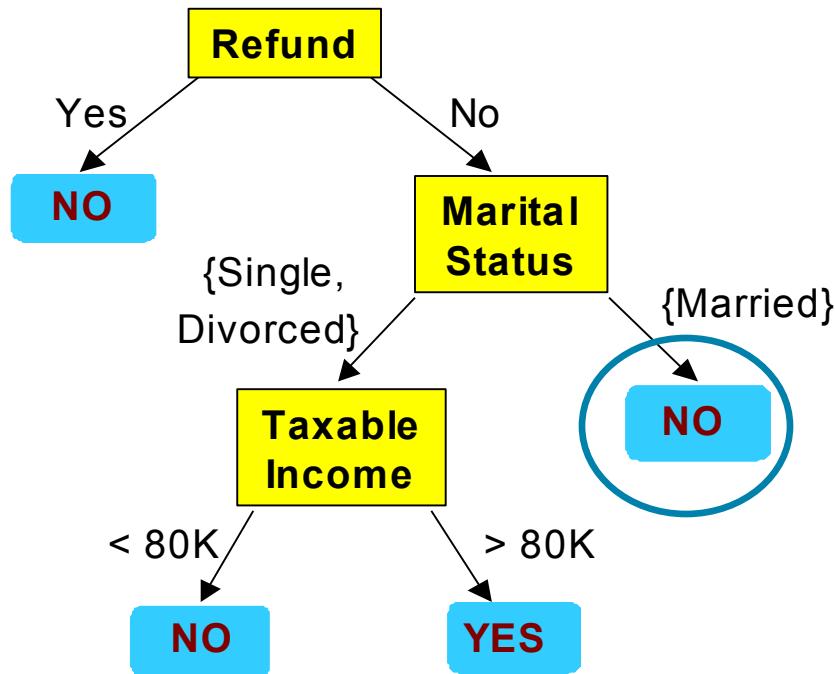
(Refund=No, Marital Status={Single,Divorced},  
Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

**Rules are mutually exclusive and exhaustive**

**Rule set contains as much information as the tree**

# Rules Can Be Simplified



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

**Initial Rule:**  $(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No}$

**Simplified Rule:**  $(\text{Status}=\text{Married}) \rightarrow \text{No}$

# Effect of Rule Simplification

---

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - ◆ Ordered rule set
    - ◆ Unordered rule set – use voting schemes
  
- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - ◆ Use a default class

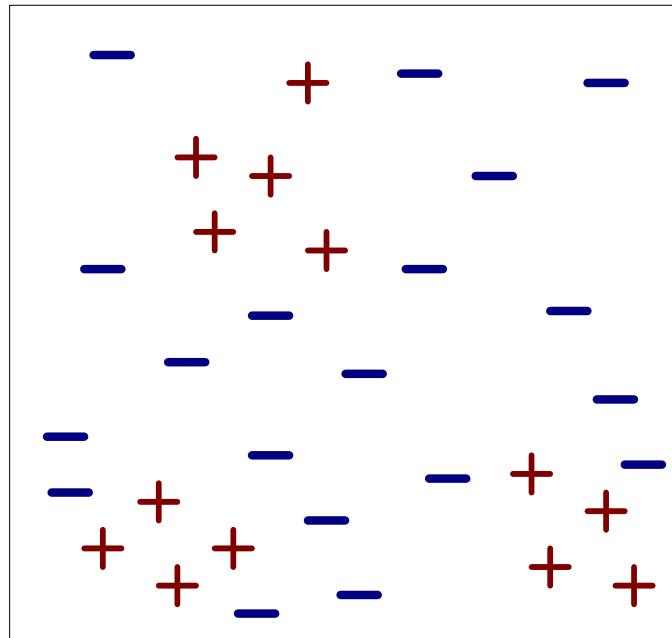
# Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

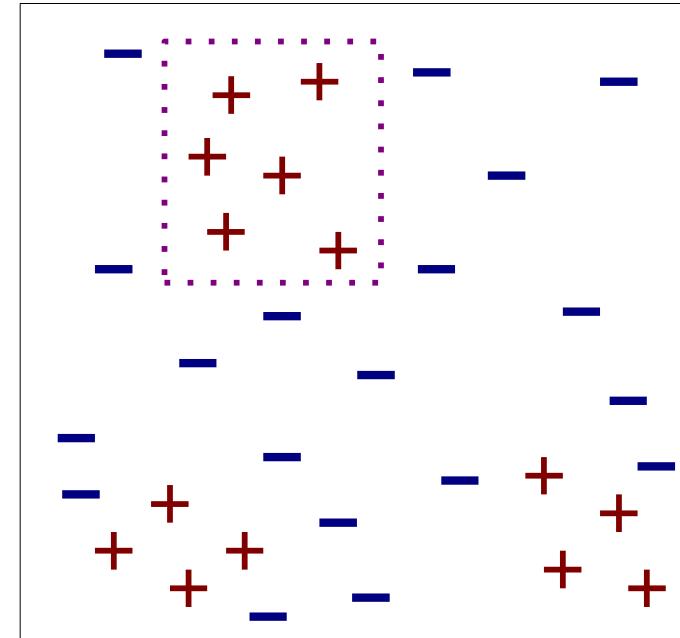
R1: (Give Birth = no)  $\wedge$  (Can Fly = yes)  $\rightarrow$  Birds  
R2: (Give Birth = no)  $\wedge$  (Live in Water = yes)  $\rightarrow$  Fishes  
R3: (Give Birth = yes)  $\wedge$  (Blood Type = warm)  $\rightarrow$  Mammals  
R4: (Give Birth = no)  $\wedge$  (Can Fly = no)  $\rightarrow$  Reptiles  
R5: (Live in Water = sometimes)  $\rightarrow$  Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

# Example of Sequential Covering

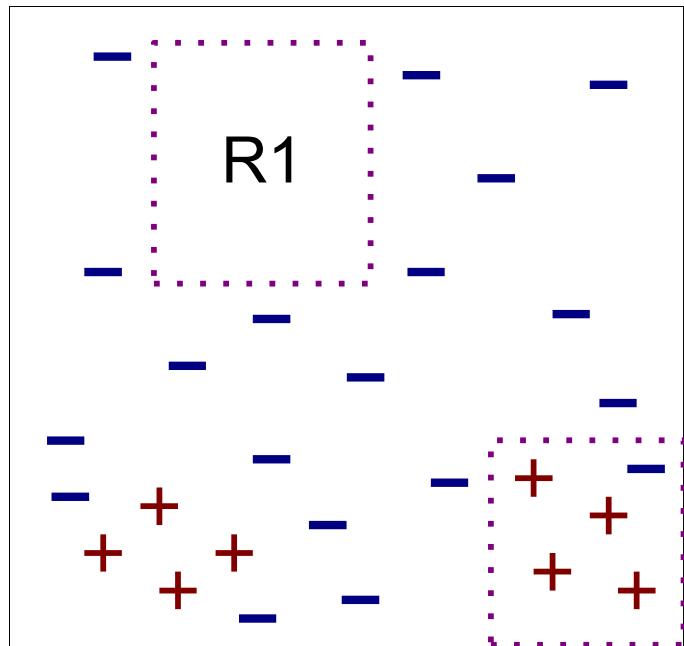


(i) Original Data

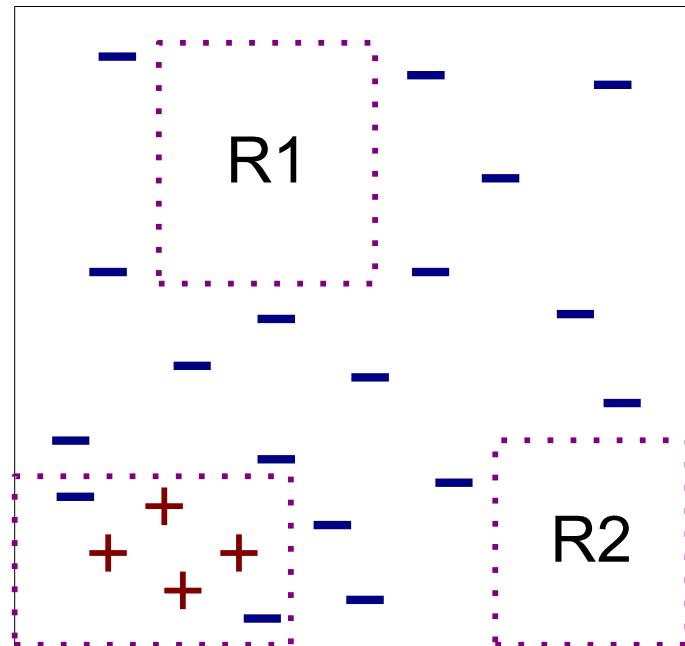


(ii) Step 1

# Example of Sequential Covering...



(iii) Step 2



(iv) Step 3

# Aspects of Sequential Covering

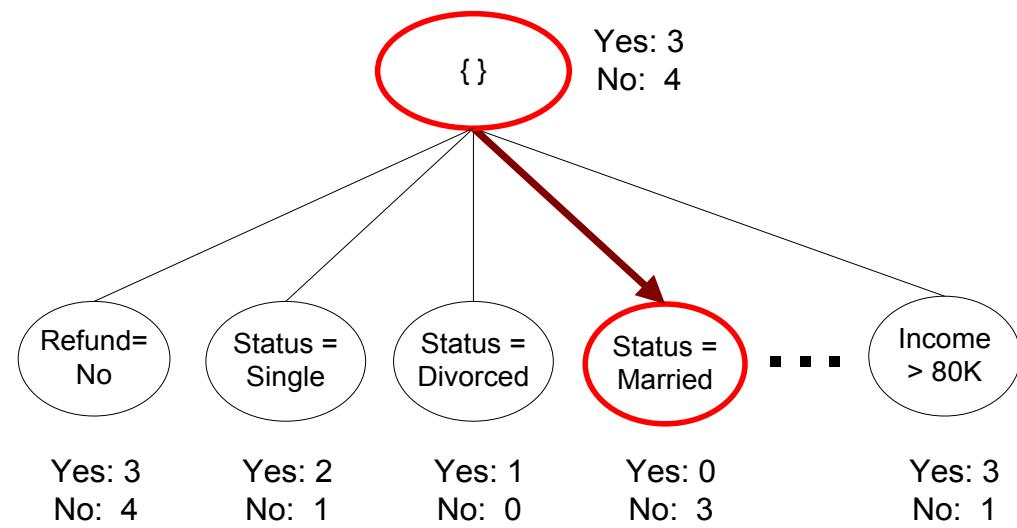
---

---

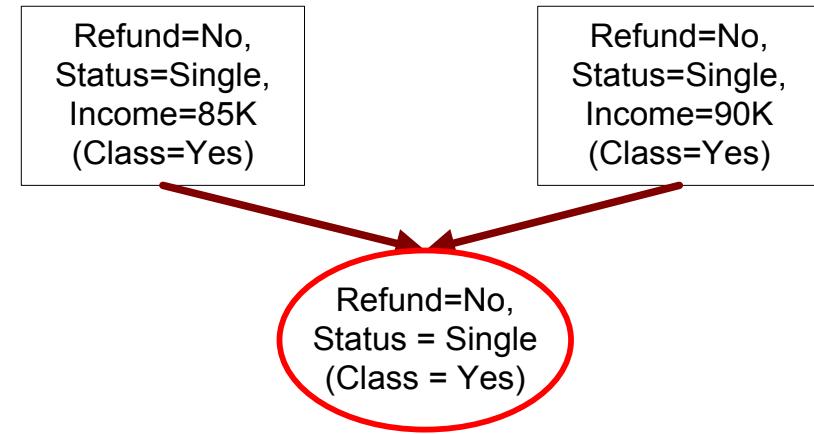
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

# Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

# Rule Growing (Examples)

---

- CN2 Algorithm:

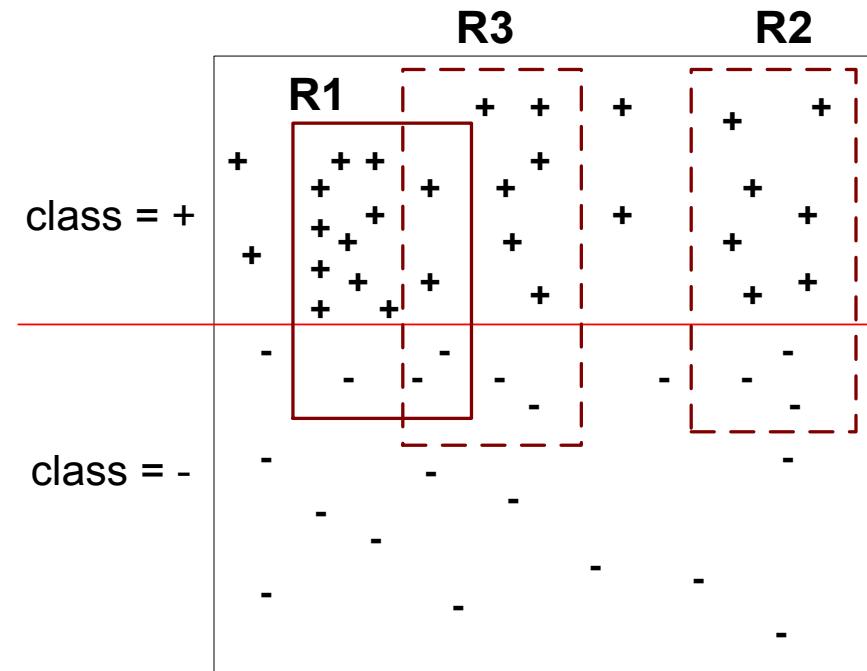
- Start from an empty conjunct: {}
- Add conjuncts that minimizes the entropy measure: {A}, {A,B}, ...
- Determine the rule consequent by taking majority class of instances covered by the rule

- RIPPER Algorithm:

- Start from an empty rule: {} => class
- Add conjuncts that maximizes FOIL's information gain measure:
  - ◆ R0: {} => class (initial rule)
  - ◆ R1: {A} => class (rule after adding conjunct)
  - ◆  $\text{Gain}(R0, R1) = t [ \log(p1/(p1+n1)) - \log(p0/(p0 + n0)) ]$
  - ◆ where t: number of positive instances covered by both R0 and R1  
p0: number of positive instances covered by R0  
n0: number of negative instances covered by R0  
p1: number of positive instances covered by R1  
n1: number of negative instances covered by R1

# Instance Elimination

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
  - Ensure that the next rule is different
- Why do we remove negative instances?
  - Prevent underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram



# Stopping Criterion and Rule Pruning

---

- Stopping criterion
  - Compute the gain
  - If gain is not significant, discard the new rule
- Rule Pruning
  - Similar to post-pruning of decision trees
  - Reduced Error Pruning:
    - ◆ Remove one of the conjuncts in the rule
    - ◆ Compare error rate on validation set before and after pruning
    - ◆ If error improves, prune the conjunct

# Summary of Direct Method

---

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

# Advantages of Rule-Based Classifiers

---

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

# Instance-Based Classifiers

Set of Stored Cases

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

Atr1	.....	AtrN

# Instance Based Classifiers

---

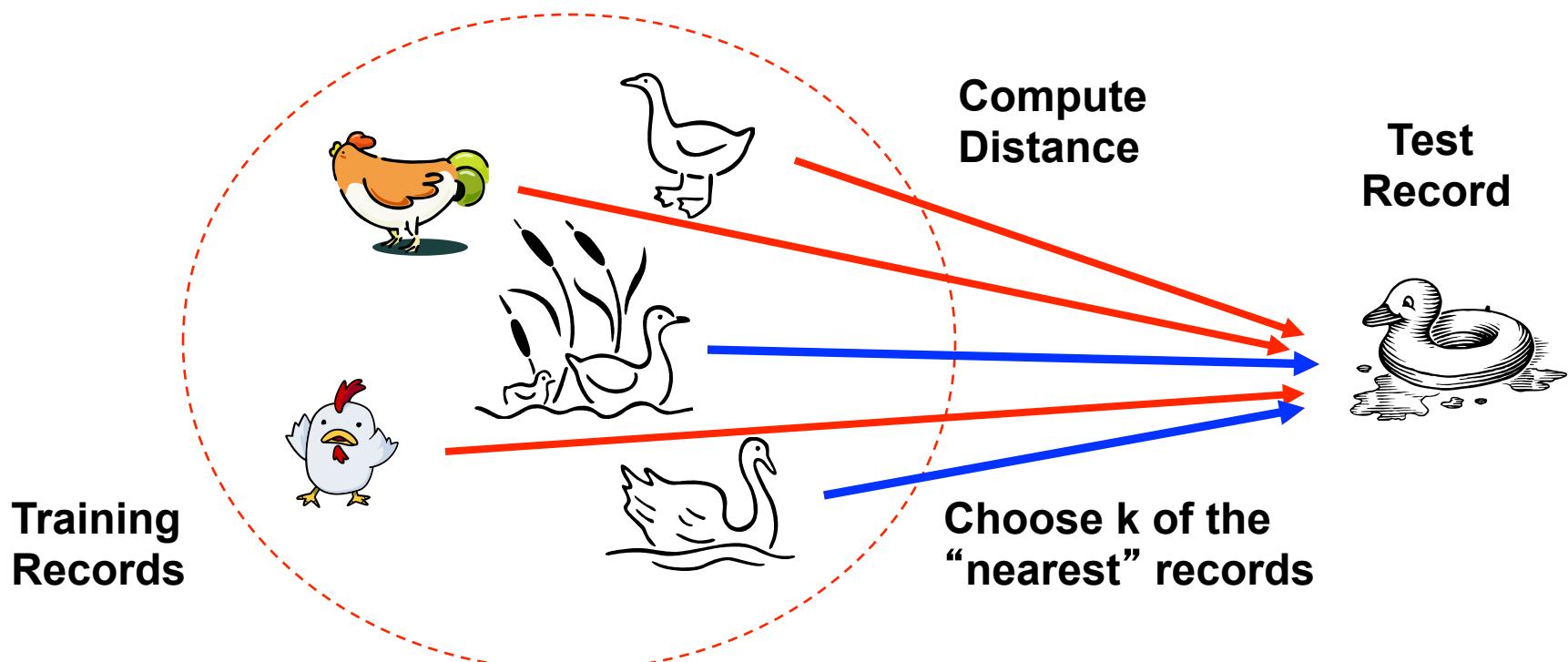
- Examples:

- Rote-learner
    - ◆ Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - ◆ Uses  $k$  “closest” points (nearest neighbors) for performing classification

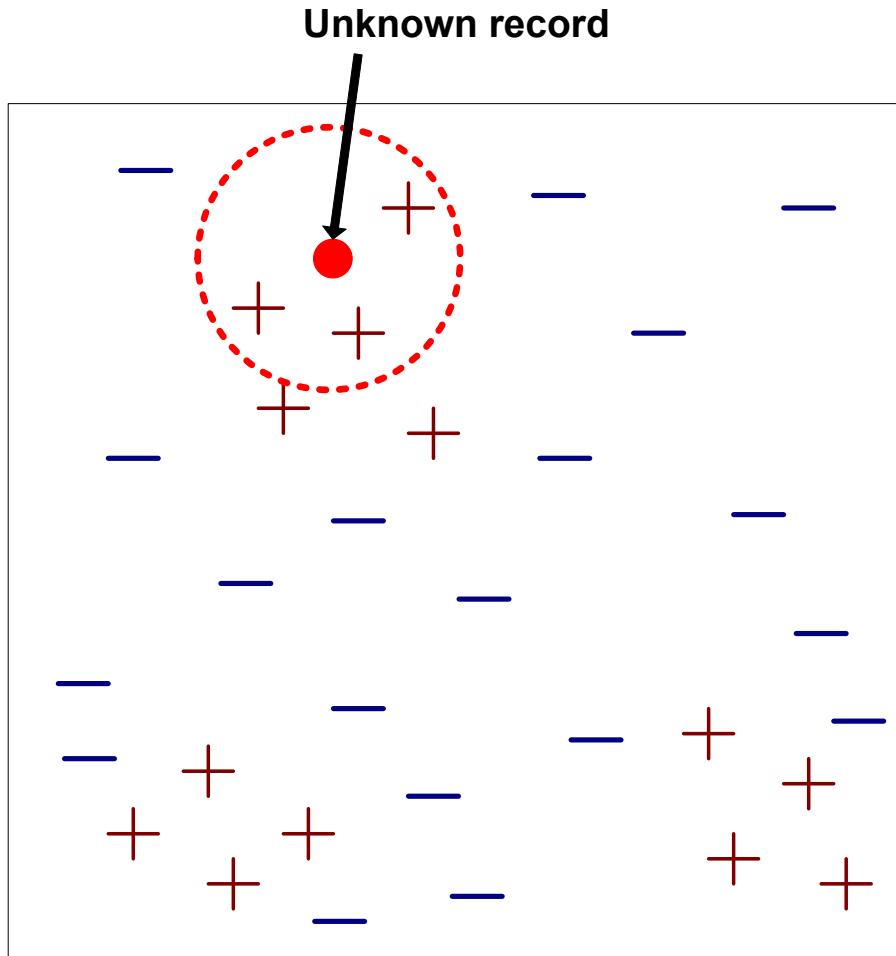
# Nearest Neighbor Classifiers

- Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck

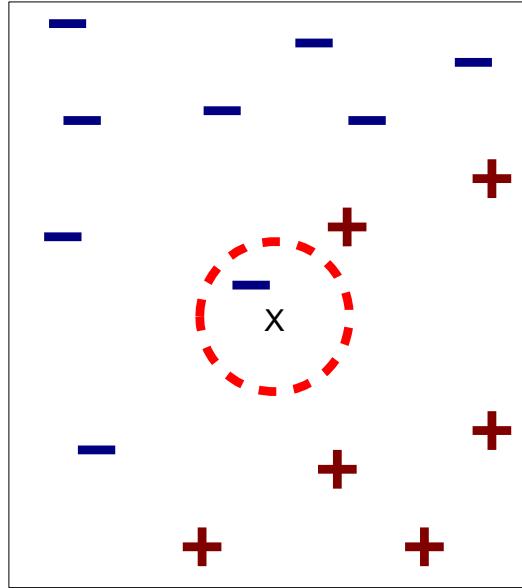


# Nearest-Neighbor Classifiers

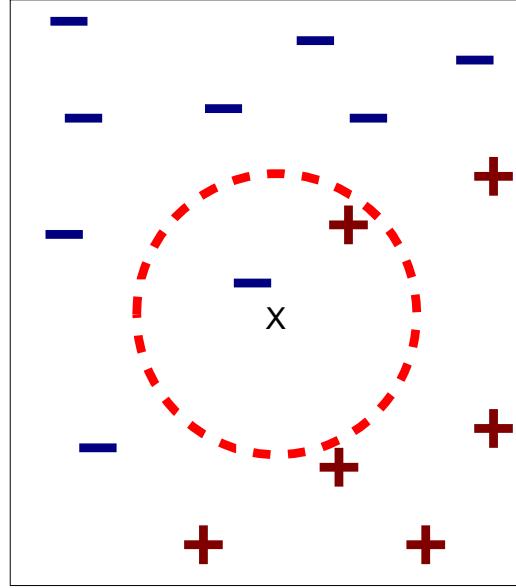


- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of  $k$ , the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify  $k$  nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

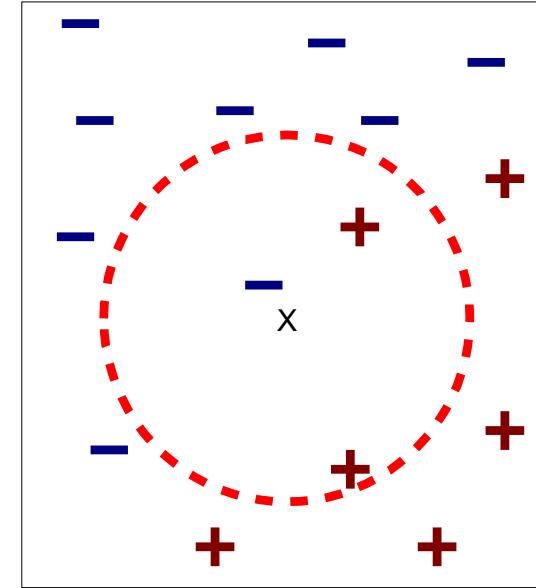
# Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

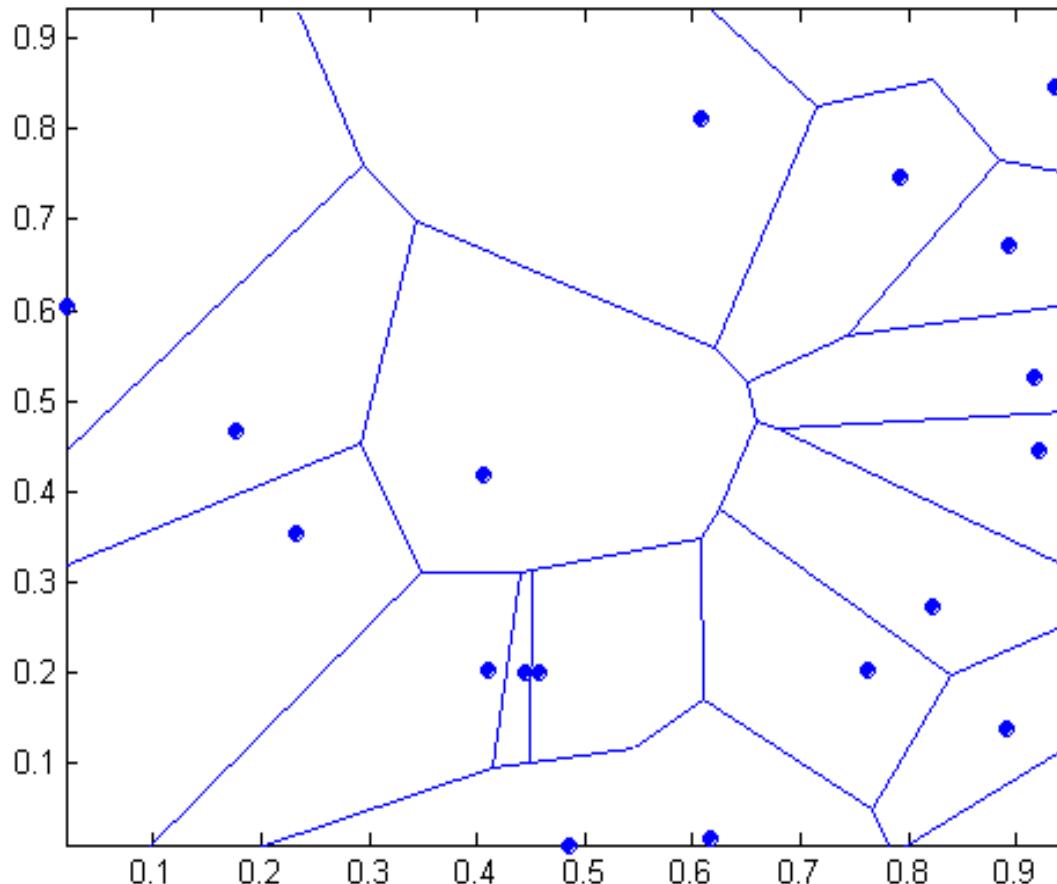


(c) 3-nearest neighbor

K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$

# 1 nearest-neighbor

## Voronoi Diagram



# Nearest Neighbor Classification

---

- Compute distance between two points:
  - Euclidean distance

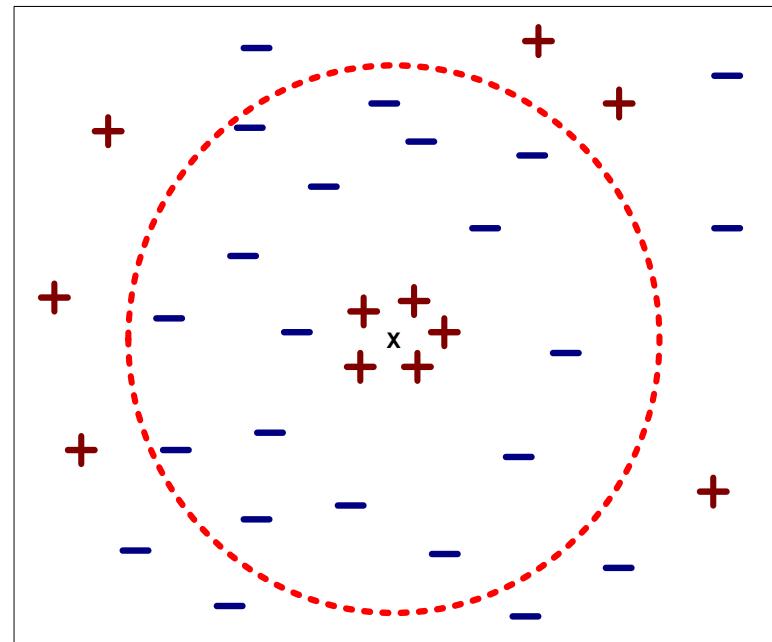
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - ◆ weight factor,  $w = 1/d^2$

# Nearest Neighbor Classification...

- Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification...

---

- Scaling issues

- Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example:
  - ◆ height of a person may vary from 1.5m to 1.8m
  - ◆ weight of a person may vary from 90lb to 300lb
  - ◆ income of a person may vary from \$10K to \$1M

# Nearest Neighbor Classification...

- Problem with Euclidean measure:
  - High dimensional data
    - ◆ curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 0

vs

0 1 1 1 1 1 1 1 1 1 1

$d = 1.4142$

1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1

$d = 1.4142$

- ◆ Solution: Normalize the vectors to unit length

# Nearest neighbor Classification...

---

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

# Example: PEBLS

---

- PEBLS: Parallel Exemplar-Based Learning System (Cost & Salzberg)
  - Works with both continuous and nominal features
    - ◆ For nominal features, distance between two nominal values is computed using modified value difference metric (MVDM)
  - Each record is assigned a weight factor
  - Number of nearest neighbor,  $k = 1$

# Example: PEBLS

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Distance between nominal attribute values:

$$d(\text{Single}, \text{Married})$$

$$= | 2/4 - 0/4 | + | 2/4 - 4/4 | = 1$$

$$d(\text{Single}, \text{Divorced})$$

$$= | 2/4 - 1/2 | + | 2/4 - 1/2 | = 0$$

$$d(\text{Married}, \text{Divorced})$$

$$= | 0/4 - 1/2 | + | 4/4 - 1/2 | = 1$$

$$d(\text{Refund}=\text{Yes}, \text{Refund}=\text{No})$$

$$= | 0/3 - 3/7 | + | 3/3 - 4/7 | = 6/7$$

Class	Marital Status		
	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

Class	Refund	
	Yes	No
Yes	0	3
No	3	4

$$d(V_1, V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|$$

# Example: PEBLS

Tid	Refund	Marital Status	Taxable Income	Cheat
X	Yes	Single	125K	No
Y	No	Married	100K	No

Distance between record X and record Y:

$$\Delta(X, Y) = w_X w_Y \sum_{i=1}^d d(X_i, Y_i)^2$$

where:

$$w_X = \frac{\text{Number of times X is used for prediction}}{\text{Number of times X predicts correctly}}$$

$w_X \approx 1$  if X makes accurate prediction most of the time

$w_X > 1$  if X is not reliable for making predictions

# Bayes Classifier

---

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C | A) = \frac{P(A, C)}{P(A)}$$

$$P(A | C) = \frac{P(A, C)}{P(C)}$$

- Bayes theorem:

$$P(C | A) = \frac{P(A | C)P(C)}{P(A)}$$

# Example of Bayes Theorem

---

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M | S) = \frac{P(S | M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

# Bayesian Classifiers

---

- Consider each attribute and class label as random variables
- Given a record with attributes  $(A_1, A_2, \dots, A_n)$ 
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes  $P(C | A_1, A_2, \dots, A_n)$
- Can we estimate  $P(C | A_1, A_2, \dots, A_n)$  directly from data?

# Bayesian Classifiers

---

- Approach:

- compute the posterior probability  $P(C | A_1, A_2, \dots, A_n)$  for all values of C using the Bayes theorem

$$P(C | A_1 A_2 \dots A_n) = \frac{P(A_1 A_2 \dots A_n | C) P(C)}{P(A_1 A_2 \dots A_n)}$$

- Choose value of C that maximizes  $P(C | A_1, A_2, \dots, A_n)$
  - Equivalent to choosing value of C that maximizes  $P(A_1, A_2, \dots, A_n | C) P(C)$

- How to estimate  $P(A_1, A_2, \dots, A_n | C)$ ?

# Naïve Bayes Classifier

---

- Assume independence among attributes  $A_i$  when class is given:
  - $P(A_1, A_2, \dots, A_n | C) = P(A_1| C_j) P(A_2| C_j) \dots P(A_n| C_j)$
  - Can estimate  $P(A_i| C_j)$  for all  $A_i$  and  $C_j$ .
  - New point is classified to  $C_j$  if  $P(C_j) \prod P(A_i| C_j)$  is maximal.

# How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class:  $P(C) = N_c/N$

- e.g.,  $P(\text{No}) = 7/10$ ,  
 $P(\text{Yes}) = 3/10$

- For discrete attributes:

$$P(A_i | C_k) = |A_{ik}| / N_{C_k}$$

- where  $|A_{ik}|$  is number of instances having attribute  $A_i$  and belongs to class  $C_k$

- Examples:

$$P(\text{Status}=\text{Married}|\text{No}) = 4/7$$
$$P(\text{Refund}=\text{Yes}|\text{Yes})=0$$

# How to Estimate Probabilities from Data?

---

- For continuous attributes:

- Discretize the range into bins
  - ◆ one ordinal attribute per bin
  - ◆ violates independence assumption  $\leftarrow k$
- Two-way split:  $(A < v)$  or  $(A > v)$ 
  - ◆ choose only one of the two splits as new attribute
- Probability density estimation:
  - ◆ Assume attribute follows a normal distribution
  - ◆ Use data to estimate parameters of distribution (e.g., mean and standard deviation)
  - ◆ Once probability distribution is known, can use it to estimate the conditional probability  $P(A_i|c)$

# How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each  $(A_i, c_i)$  pair

- For (Income, Class=No):

- If Class=No

- sample mean = 110

- sample variance = 2975

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

# Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

$$P(\text{Refund}=\text{Yes}|\text{No}) = 3/7$$

$$P(\text{Refund}=\text{No}|\text{No}) = 4/7$$

$$P(\text{Refund}=\text{Yes}|\text{Yes}) = 0$$

$$P(\text{Refund}=\text{No}|\text{Yes}) = 1$$

$$P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7$$

$$P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7$$

$$P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/7$$

$$P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0$$

For taxable income:

If class=No: sample mean=110 sample variance=2975

If class=Yes: sample mean=90 sample variance=25

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{ Class}=\text{No}) \times P(\text{Income}=120\text{K}|\text{ Class}=\text{No}) = 4/7 \times 4/7 \times 0.0072 = 0.0024$
- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{ Class}=\text{Yes}) \times P(\text{Married}|\text{ Class}=\text{Yes}) \times P(\text{Income}=120\text{K}|\text{ Class}=\text{Yes}) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since  $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore  $P(\text{No}|X) > P(\text{Yes}|X)$   
=> Class = No

# Naïve Bayes Classifier

---

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original : } P(A_i | C) = \frac{N_{ic}}{N_c}$$

c: number of classes

$$\text{Laplace : } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

p: prior probability

$$\text{m - estimate : } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

m: parameter

# Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A | M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A | N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$P(A|M)P(M) > P(A|N)P(N)$   
 $\Rightarrow$  Mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

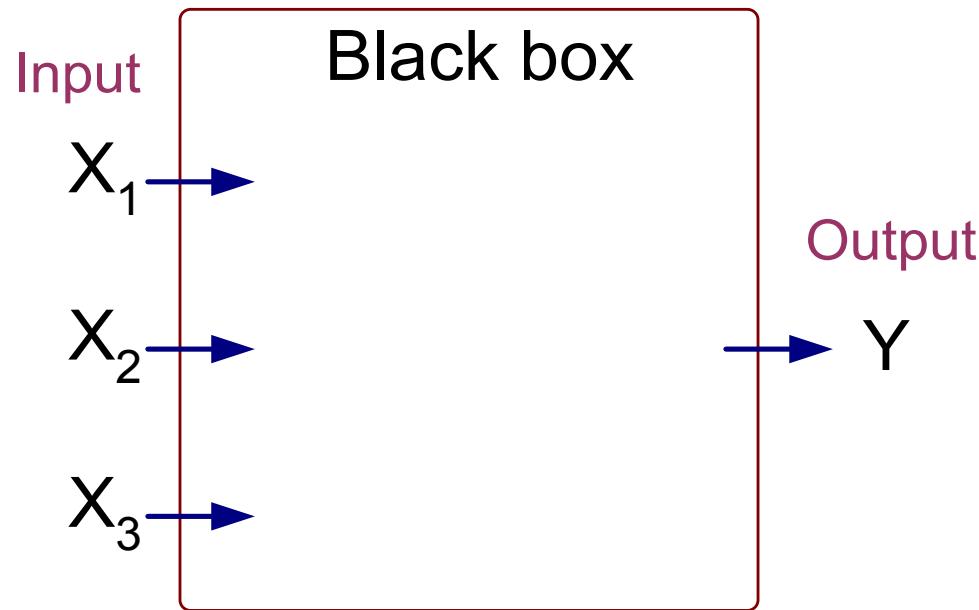
# Naïve Bayes (Summary)

---

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

# Artificial Neural Networks (ANN)

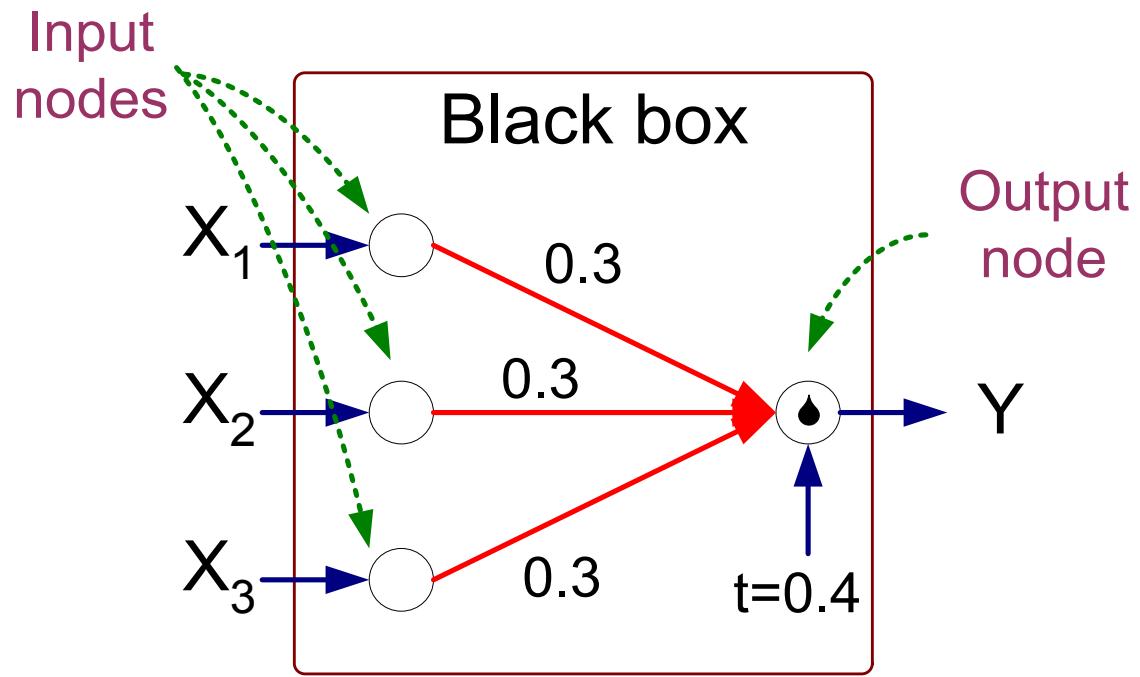
$X_1$	$X_2$	$X_3$	$Y$
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output  $Y$  is 1 if at least two of the three inputs are equal to 1.

# Artificial Neural Networks (ANN)

$X_1$	$X_2$	$X_3$	$Y$
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

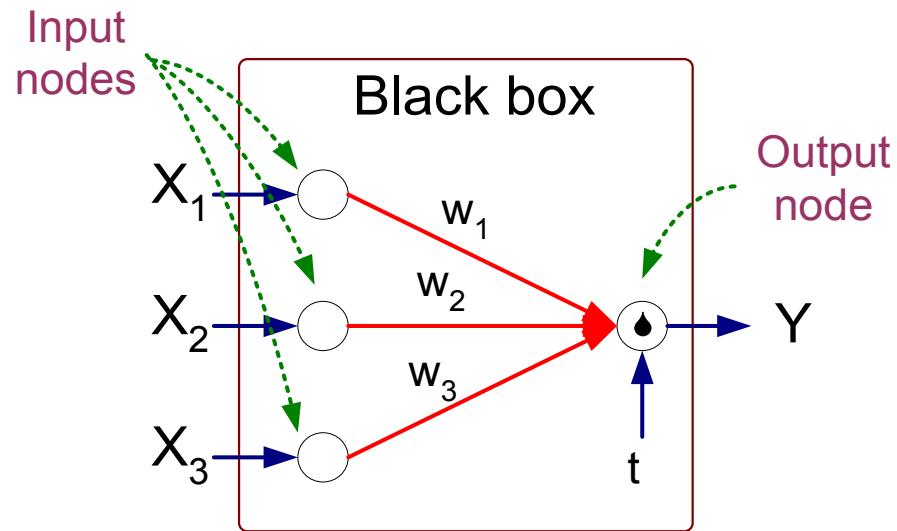


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

where  $I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$

# Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

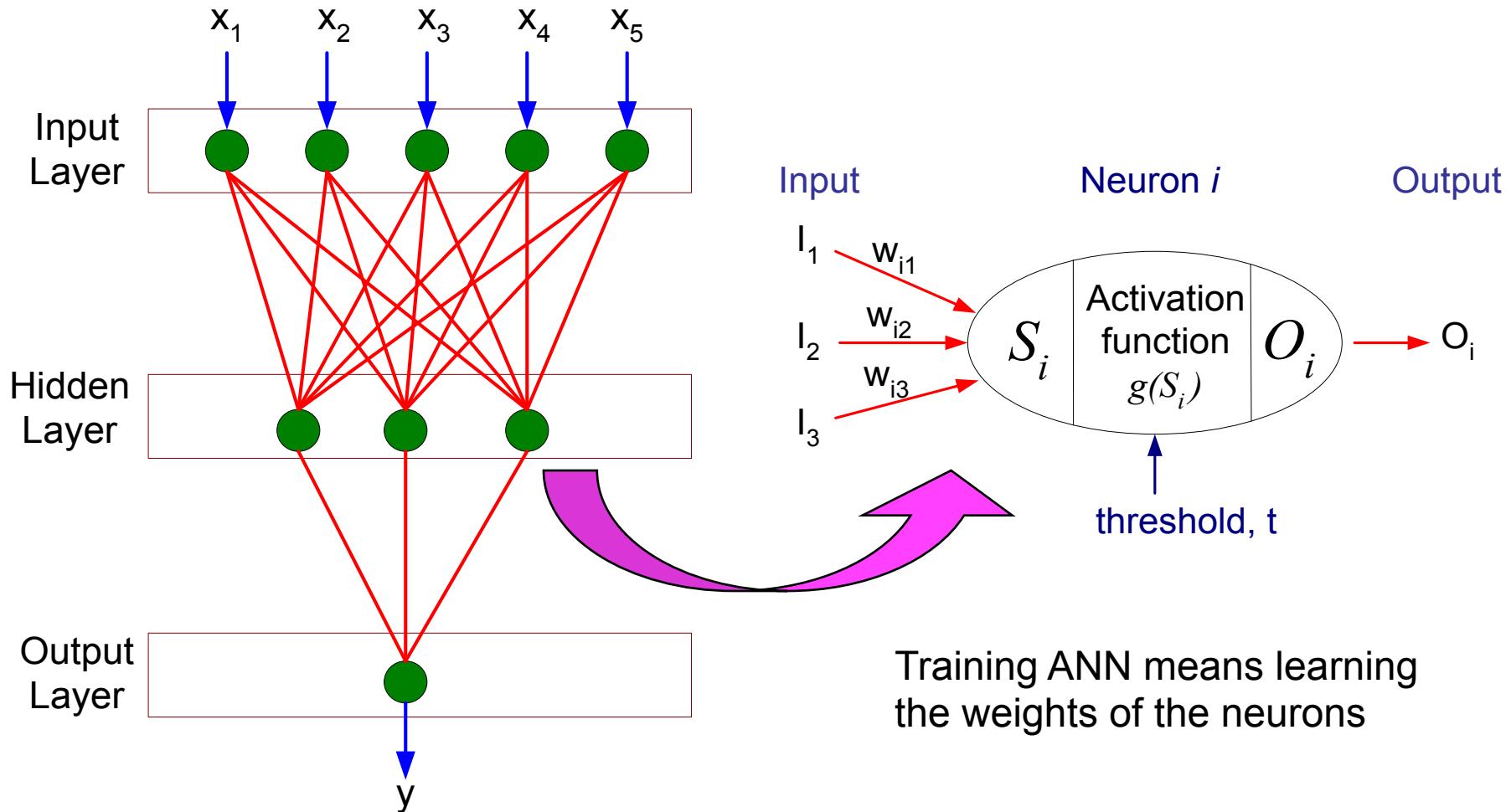


Perceptron Model

$$Y = I\left(\sum_i w_i X_i - t\right) \quad \text{or}$$

$$Y = sign\left(\sum_i w_i X_i - t\right)$$

# General Structure of ANN

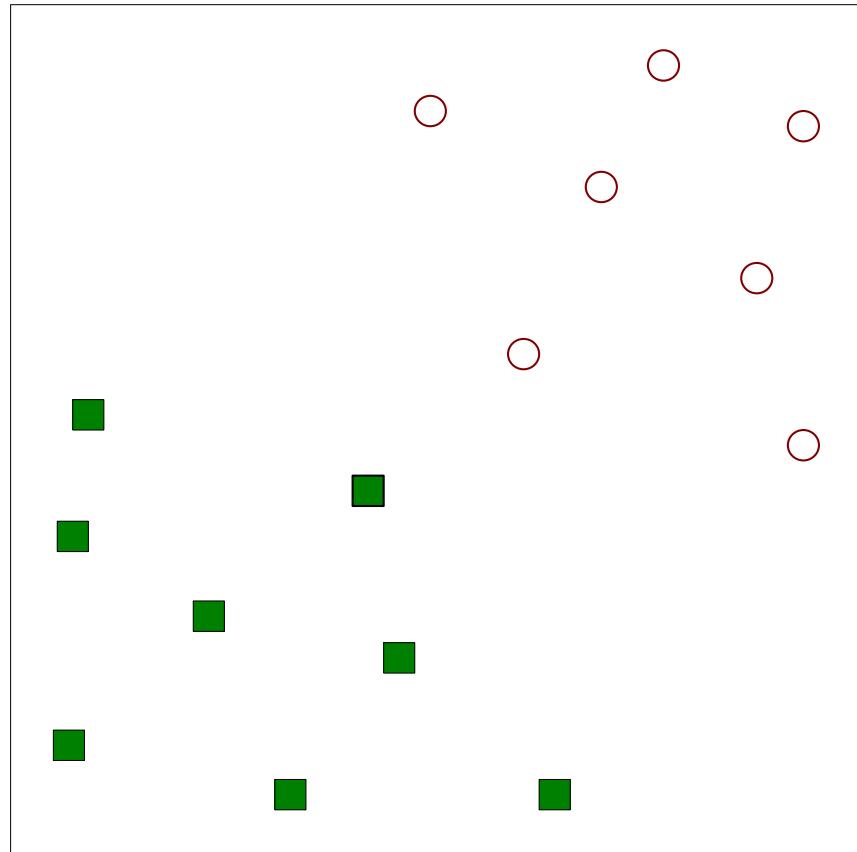


# Algorithm for learning ANN

---

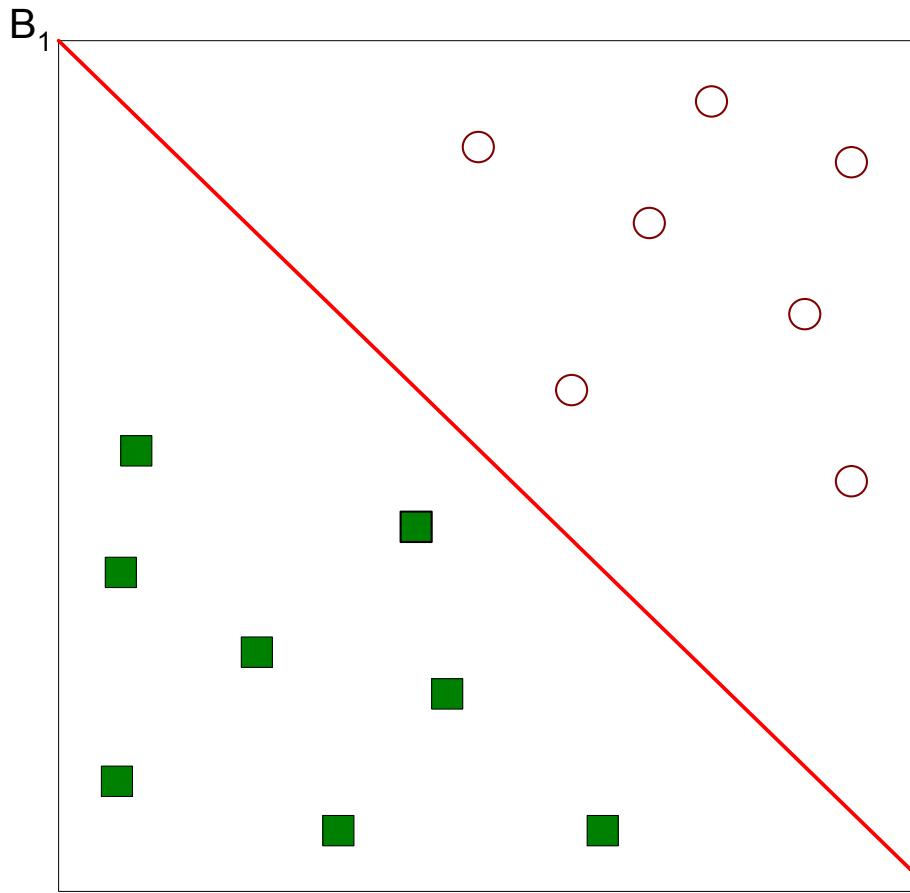
- Initialize the weights ( $w_0, w_1, \dots, w_k$ )
- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Objective function:  $E = \sum_i [Y_i - f(w_i, X_i)]^2$
  - Find the weights  $w_i$ 's that minimize the above objective function
    - ◆ e.g., backpropagation algorithm (see lecture notes)

# Support Vector Machines



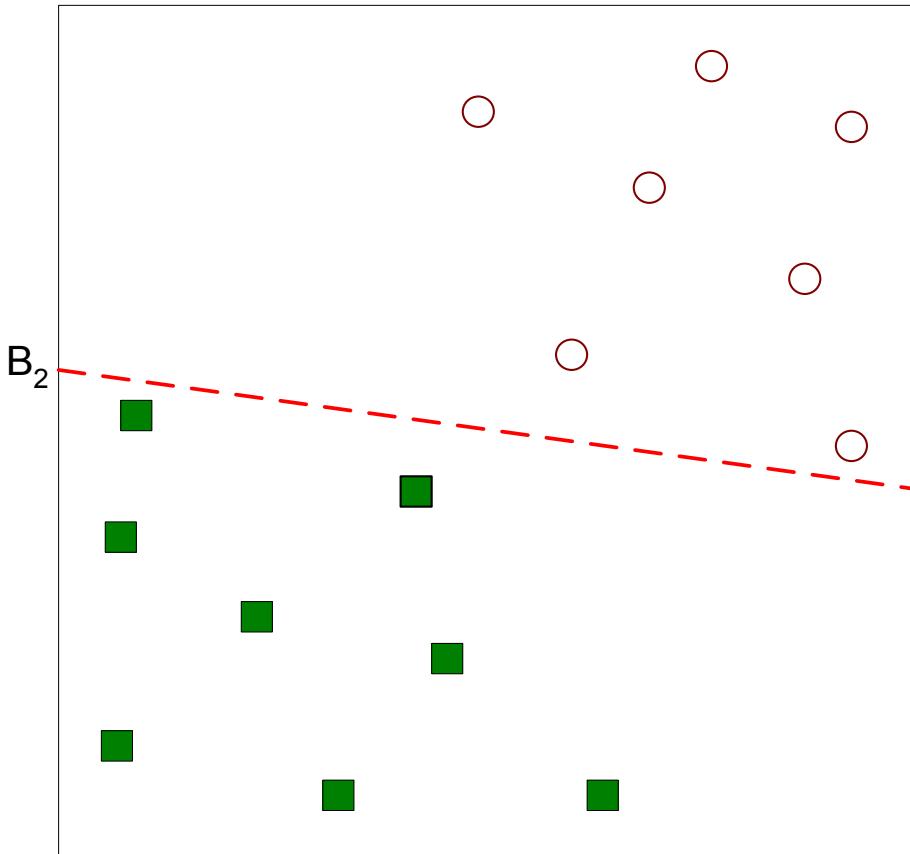
- Find a linear hyperplane (decision boundary) that will separate the data

# Support Vector Machines



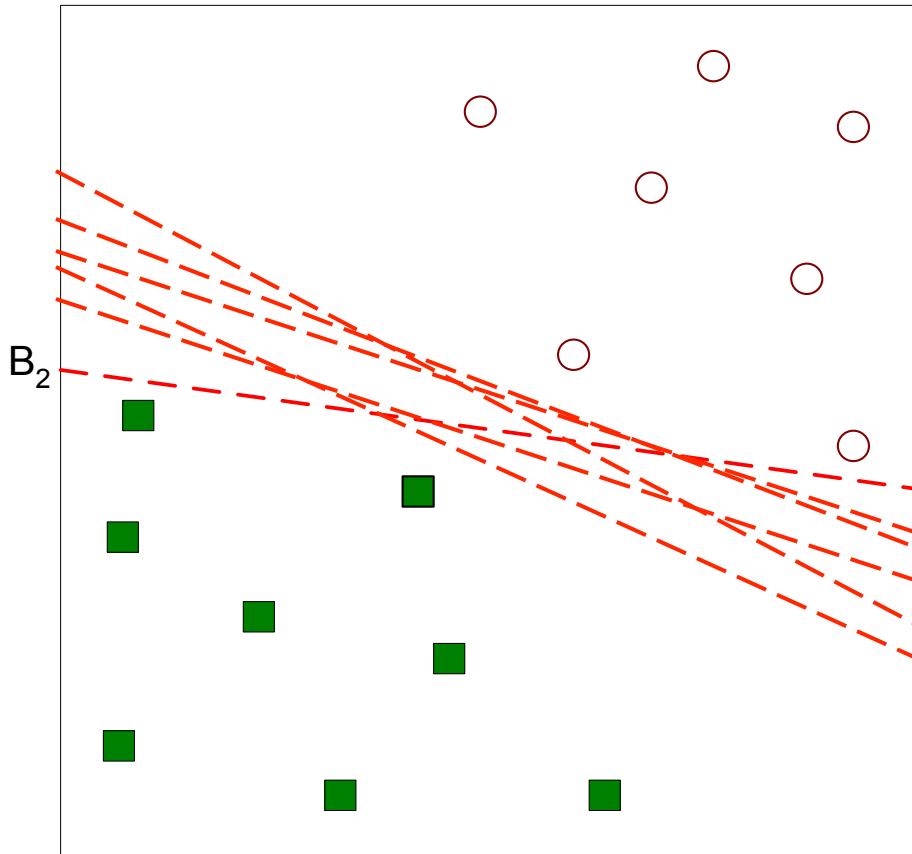
- One Possible Solution

# Support Vector Machines



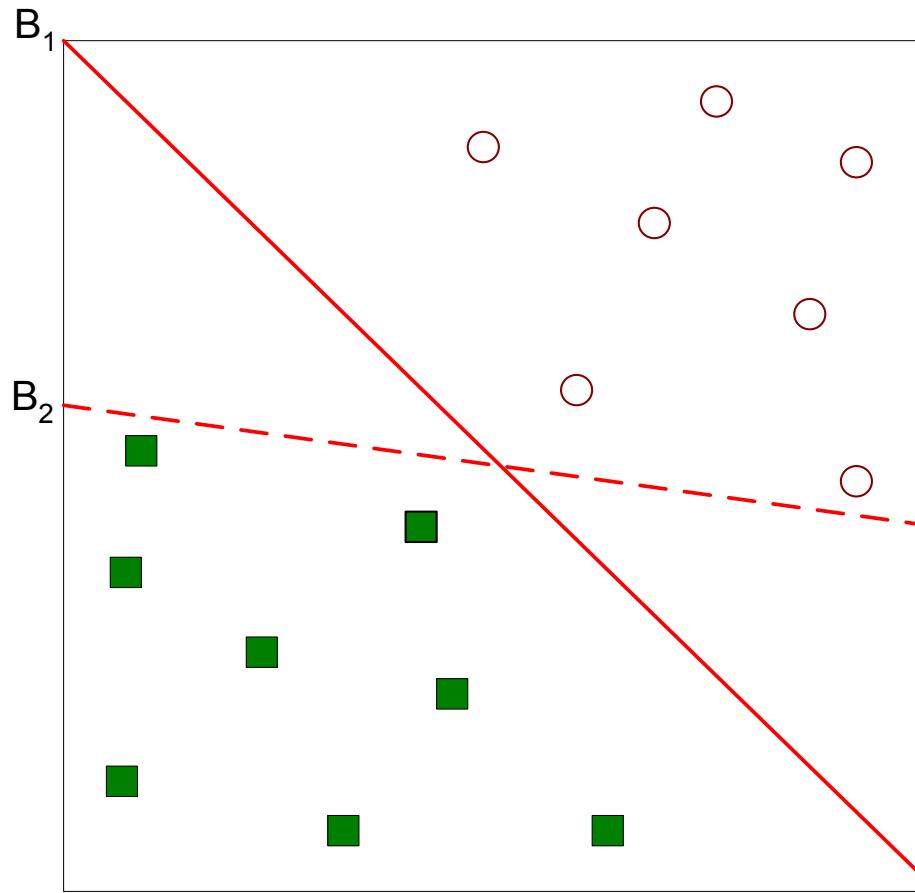
- Another possible solution

# Support Vector Machines



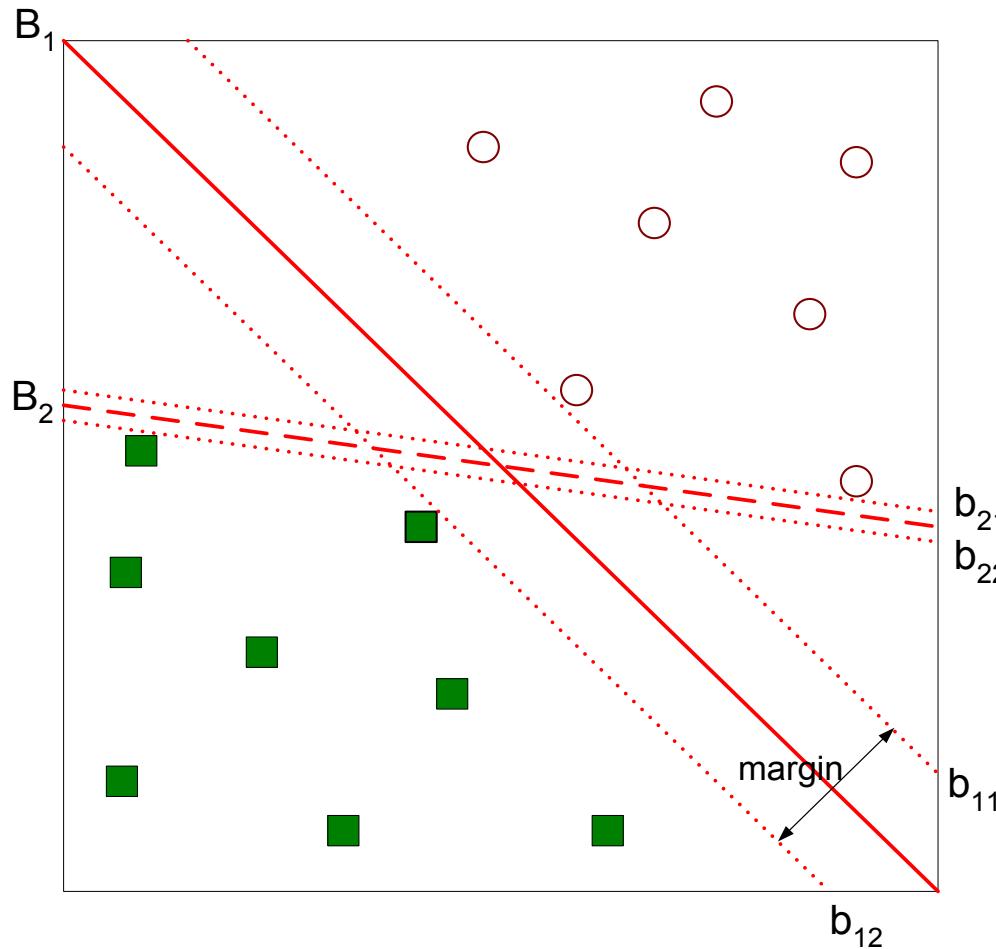
- Other possible solutions

# Support Vector Machines



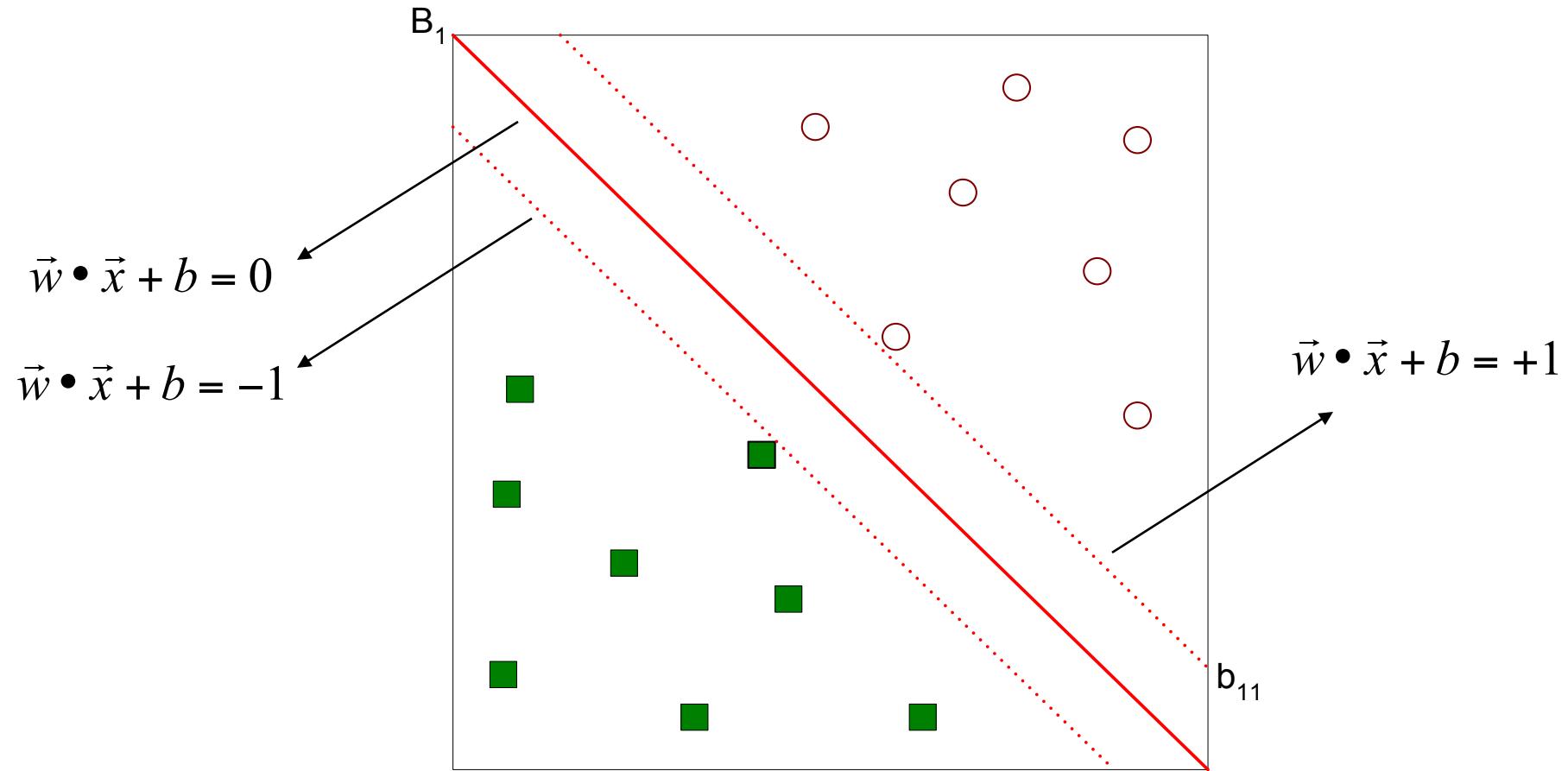
- Which one is better? B1 or B2?
- How do you define better?

# Support Vector Machines



- Find hyperplane **maximizes** the margin => B1 is better than B2

# Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

# Support Vector Machines

---

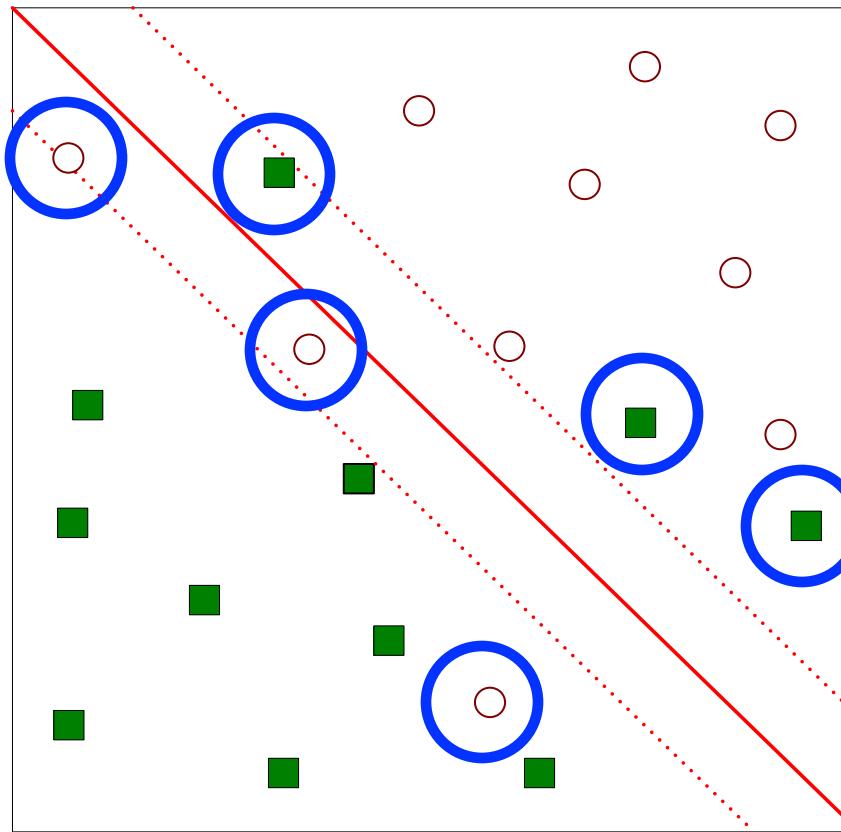
- We want to maximize: Margin =  $\frac{2}{\|\vec{w}\|^2}$ 
  - Which is equivalent to minimizing:  $L(w) = \frac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 \end{cases}$$

- ◆ This is a constrained optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)

# Support Vector Machines

- What if the problem is not linearly separable?



# Support Vector Machines

---

- What if the problem is not linearly separable?

- Introduce slack variables

- ◆ Need to minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left( \sum_{i=1}^N \xi_i^k \right)$$

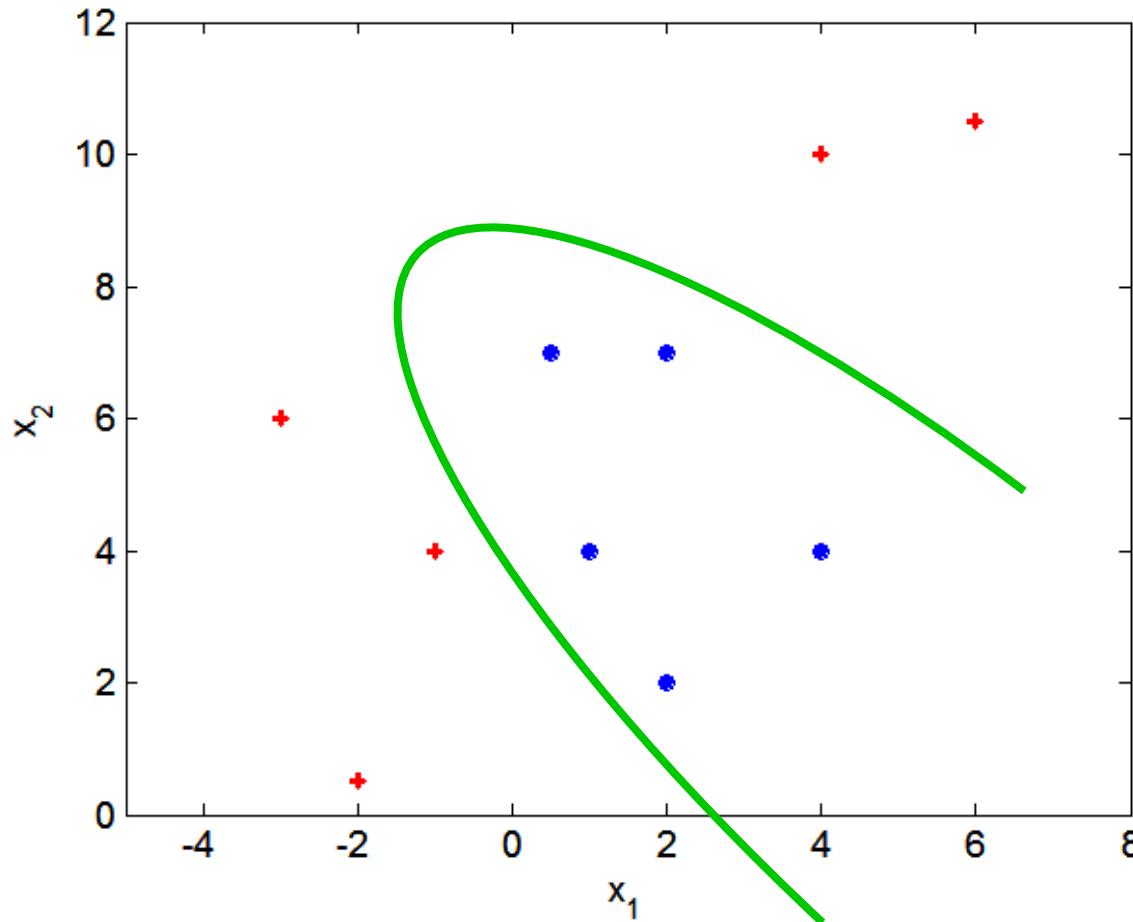
- ◆ Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

# Nonlinear Support Vector Machines

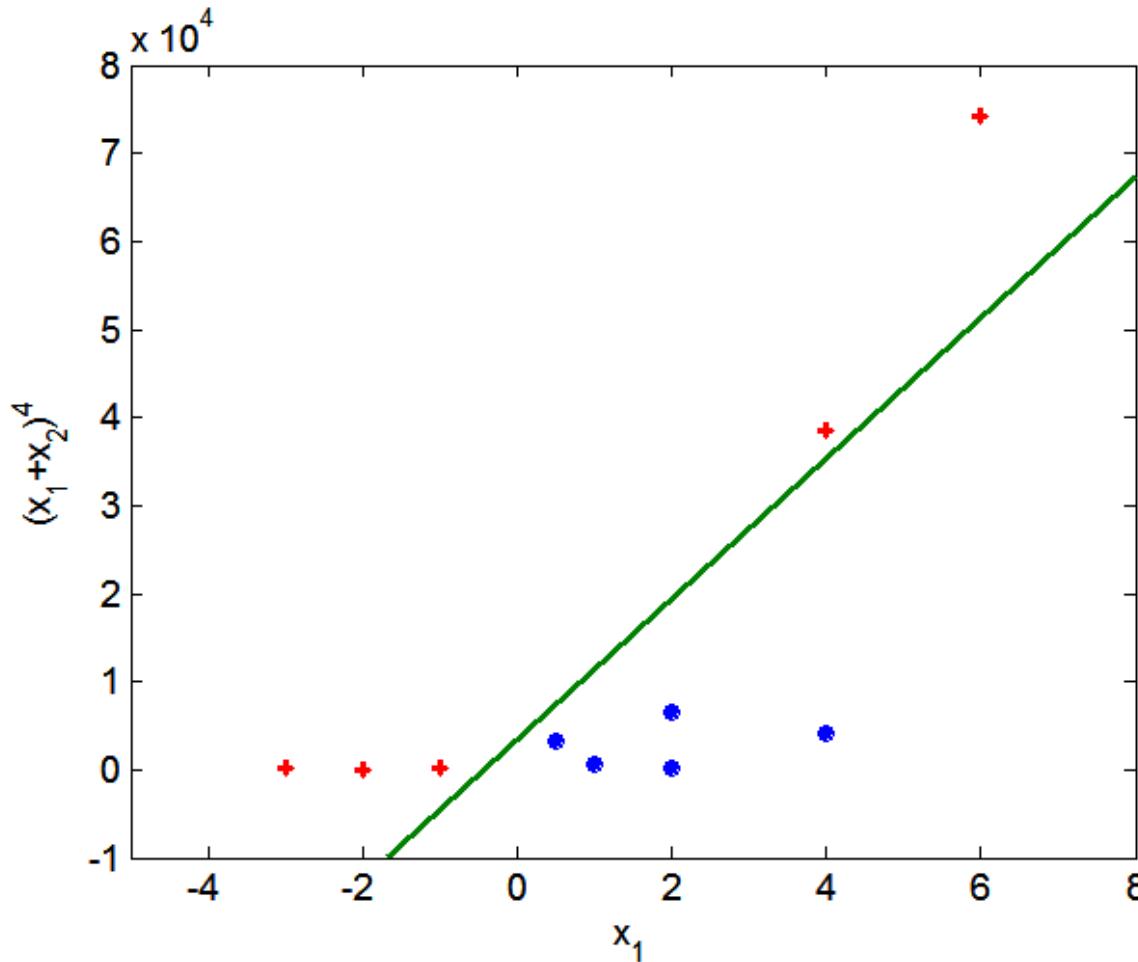
---

- What if decision boundary is not linear?



# Nonlinear Support Vector Machines

- Transform data into higher dimensional space



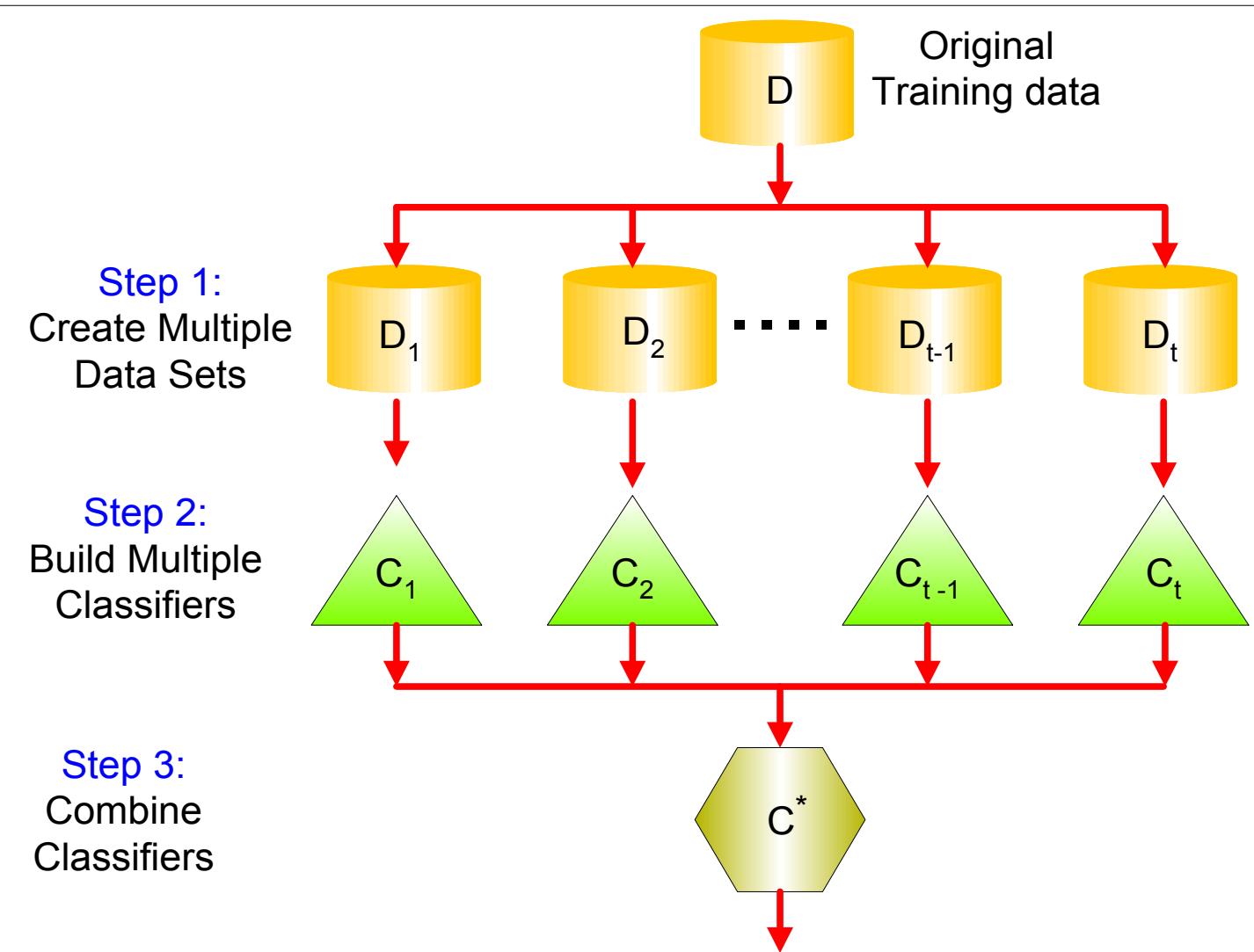
# Ensemble Methods

---

---

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

# General Idea



# Why does it work?

---

- Suppose there are 25 base classifiers
  - Each classifier has error rate,  $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1 - \varepsilon)^{25-i} = 0.06$$

# Examples of Ensemble Methods

---

---

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

# Bagging

---

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability  $(1 - 1/n)^n$  of being selected

# Boosting

---

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

# Boosting

---

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

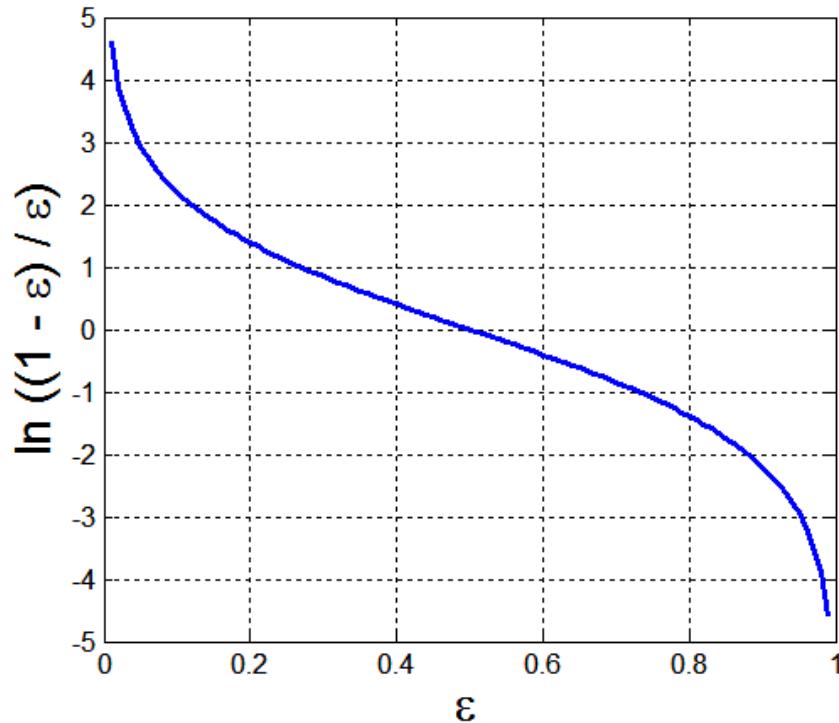
# Example: AdaBoost

- Base classifiers:  $C_1, C_2, \dots, C_T$
- Error rate:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$



# Example: AdaBoost

---

- Weight update:

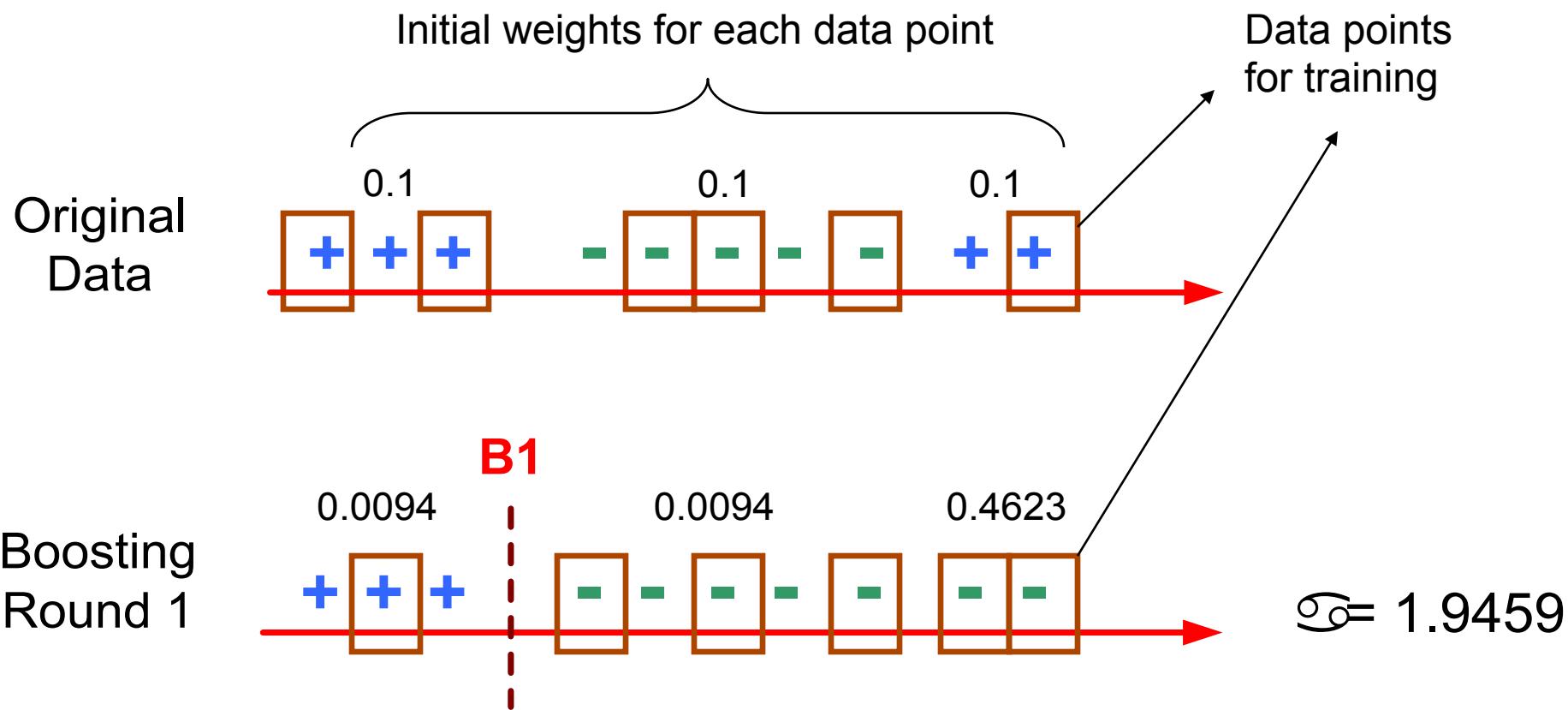
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where  $Z_j$  is the normalization factor

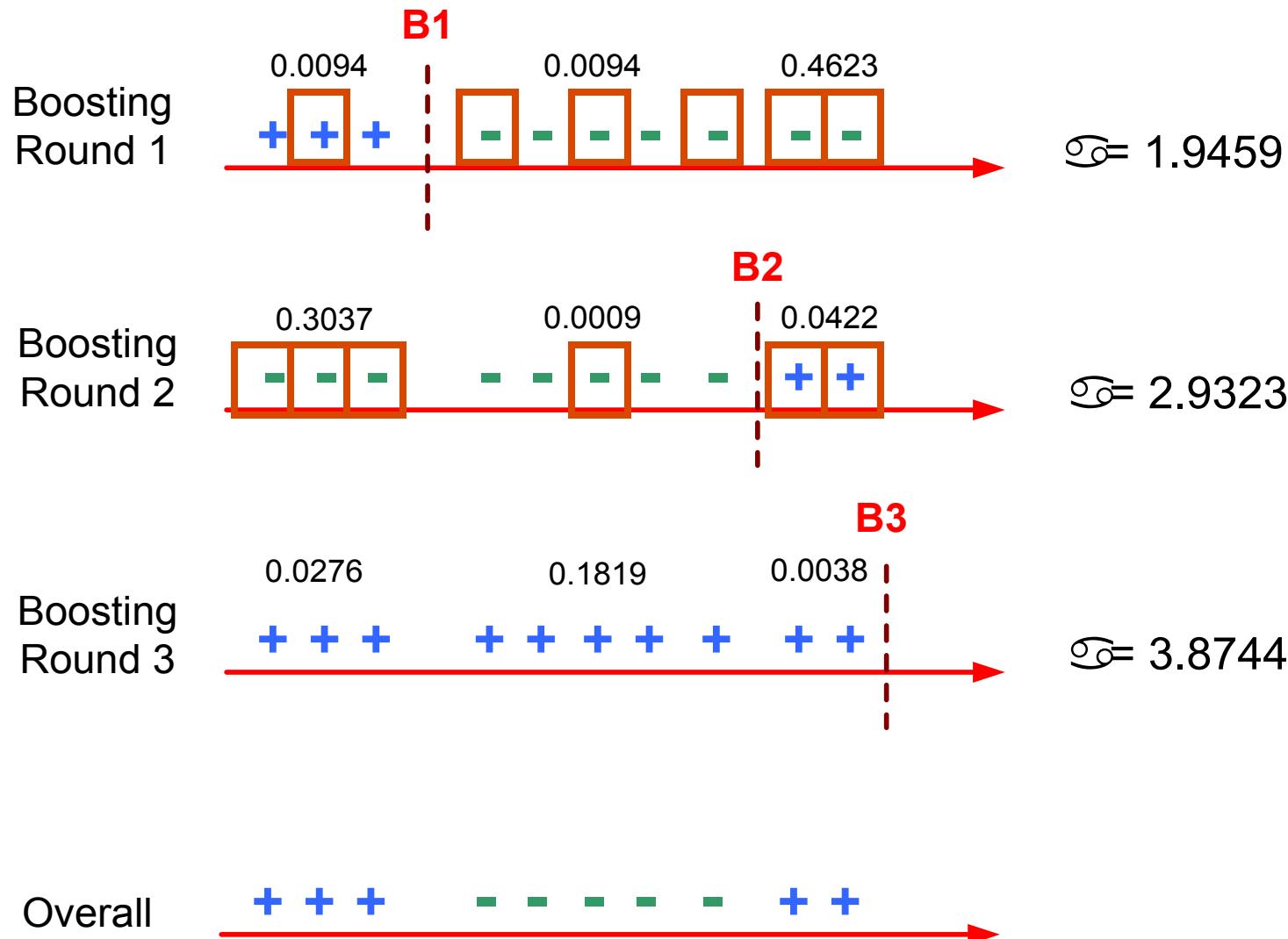
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to  $1/n$  and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

# Illustrating AdaBoost



# Illustrating AdaBoost



# Data Mining Association Analysis: Basic Concepts and Algorithms

---

---

## Lecture Notes for Chapter 6

Introduction to Data Mining

by

Tan, Steinbach, Kumar

# Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

## Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

## Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$ ,  
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\}$ ,  
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\}$ ,

Implication means co-occurrence,  
not causality!

# Definition: Frequent Itemset

## □ Itemset

- A collection of one or more items
  - ◆ Example: {Milk, Bread, Diaper}
- k-itemset
  - ◆ An itemset that contains k items

## □ Support count ( $\sigma$ )

- Frequency of occurrence of an itemset
- E.g.  $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$

## □ Support

- Fraction of transactions that contain an itemset
- E.g.  $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$

## □ Frequent Itemset

- An itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

# Definition: Association Rule

## □ Association Rule

- An implication expression of the form  $X \rightarrow Y$ , where X and Y are itemsets
- Example:  
 $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

## □ Rule Evaluation Metrics

- Support (s)
  - ◆ Fraction of transactions that contain both X and Y
- Confidence (c)
  - ◆ Measures how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

# Association Rule Mining Task

---

- Given a set of transactions  $T$ , the goal of association rule mining is to find all rules having
  - support  $\geq \text{minsup}$  threshold
  - confidence  $\geq \text{minconf}$  threshold
- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the  $\text{minsup}$  and  $\text{minconf}$  thresholds

⇒ Computationally prohibitive!

# Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

## Example of Rules:

$\{\text{Milk}, \text{Diaper}\} \rightarrow \{\text{Beer}\}$  ( $s=?$ ,  $c=?$ )  
 $\{\text{Milk}, \text{Beer}\} \rightarrow \{\text{Diaper}\}$  ( $s=?$ ,  $c=?$ )  
 $\{\text{Diaper}, \text{Beer}\} \rightarrow \{\text{Milk}\}$  ( $s=?$ ,  $c=?$ )  
 $\{\text{Beer}\} \rightarrow \{\text{Milk}, \text{Diaper}\}$  ( $s=?$ ,  $c=?$ )  
 $\{\text{Diaper}\} \rightarrow \{\text{Milk}, \text{Beer}\}$  ( $s=?$ ,  $c=?$ )  
 $\{\text{Milk}\} \rightarrow \{\text{Diaper}, \text{Beer}\}$  ( $s=?$ ,  $c=?$ )

## Observations:

- All the above rules are binary partitions of the same itemset:  
 $\{\text{Milk}, \text{Diaper}, \text{Beer}\}$
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

# Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

## Example of Rules:

$\{\text{Milk}, \text{Diaper}\} \rightarrow \{\text{Beer}\}$  ( $s=0.4, c=0.67$ )  
 $\{\text{Milk}, \text{Beer}\} \rightarrow \{\text{Diaper}\}$  ( $s=0.4, c=1.0$ )  
 $\{\text{Diaper}, \text{Beer}\} \rightarrow \{\text{Milk}\}$  ( $s=0.4, c=0.67$ )  
 $\{\text{Beer}\} \rightarrow \{\text{Milk}, \text{Diaper}\}$  ( $s=0.4, c=0.67$ )  
 $\{\text{Diaper}\} \rightarrow \{\text{Milk}, \text{Beer}\}$  ( $s=0.4, c=0.5$ )  
 $\{\text{Milk}\} \rightarrow \{\text{Diaper}, \text{Beer}\}$  ( $s=0.4, c=0.5$ )

## Observations:

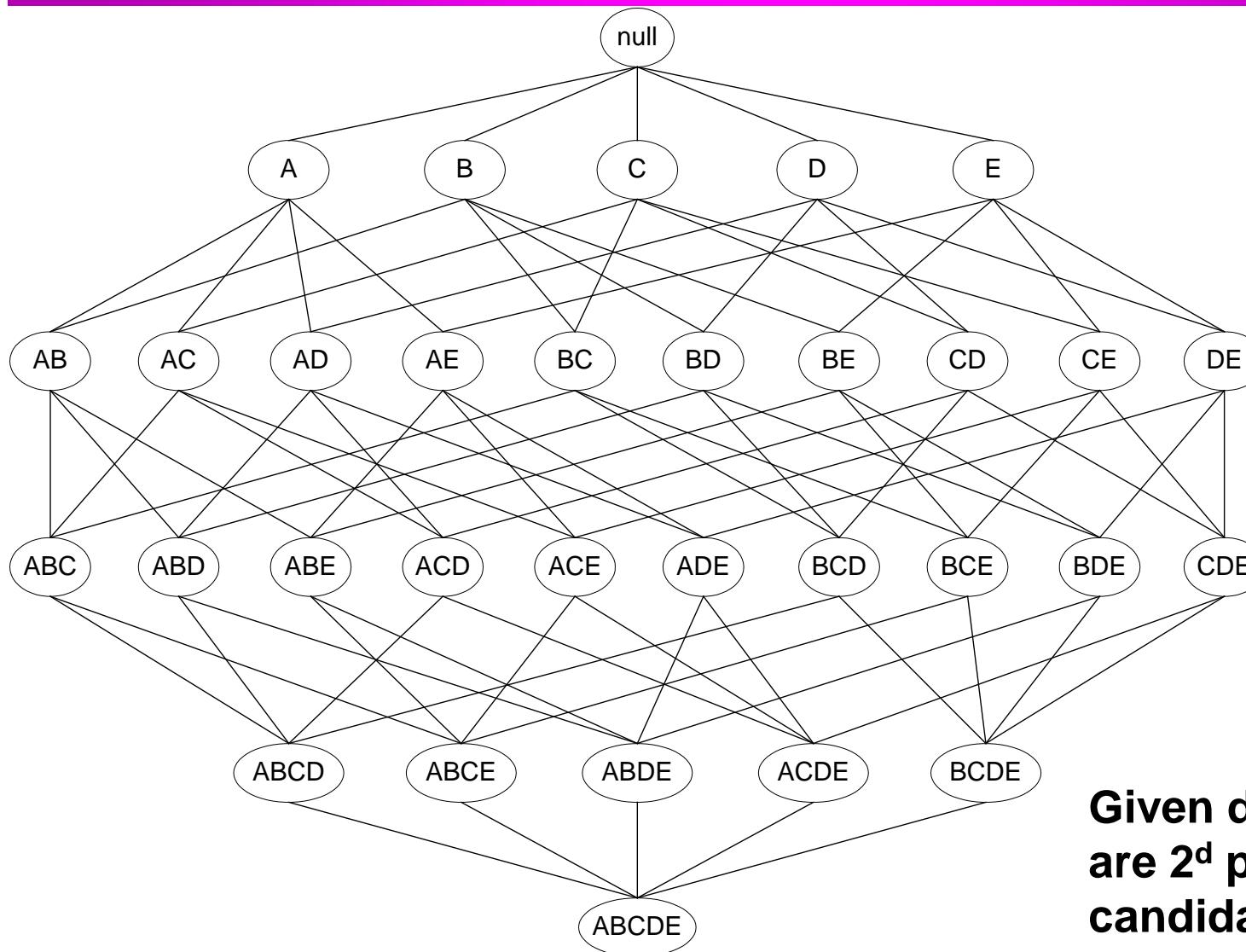
- All the above rules are binary partitions of the same itemset:  
 $\{\text{Milk}, \text{Diaper}, \text{Beer}\}$
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

# Mining Association Rules

---

- Two-step approach:
  1. Frequent Itemset Generation
    - Generate all itemsets whose support  $\geq \text{minsup}$
  2. Rule Generation
    - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

# Frequent Itemset Generation

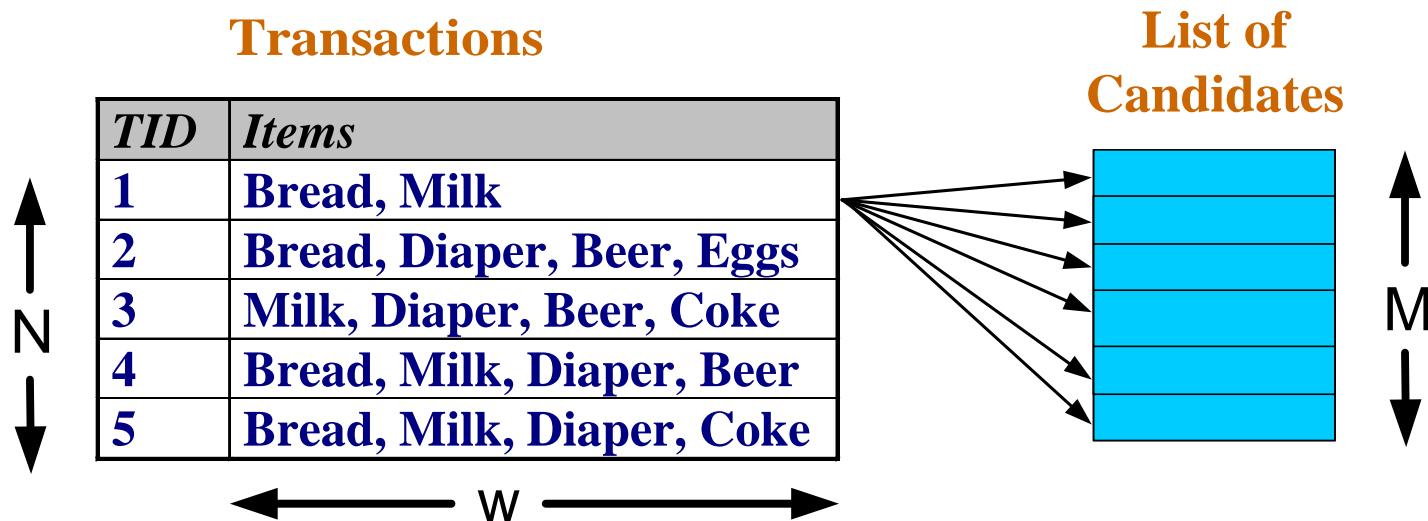


**Given  $d$  items, there  
are  $2^d$  possible  
candidate itemsets**

# Frequent Itemset Generation

## □ Brute-force approach:

- Each itemset in the lattice is a **candidate** frequent itemset
- Count the support of each candidate by scanning the database

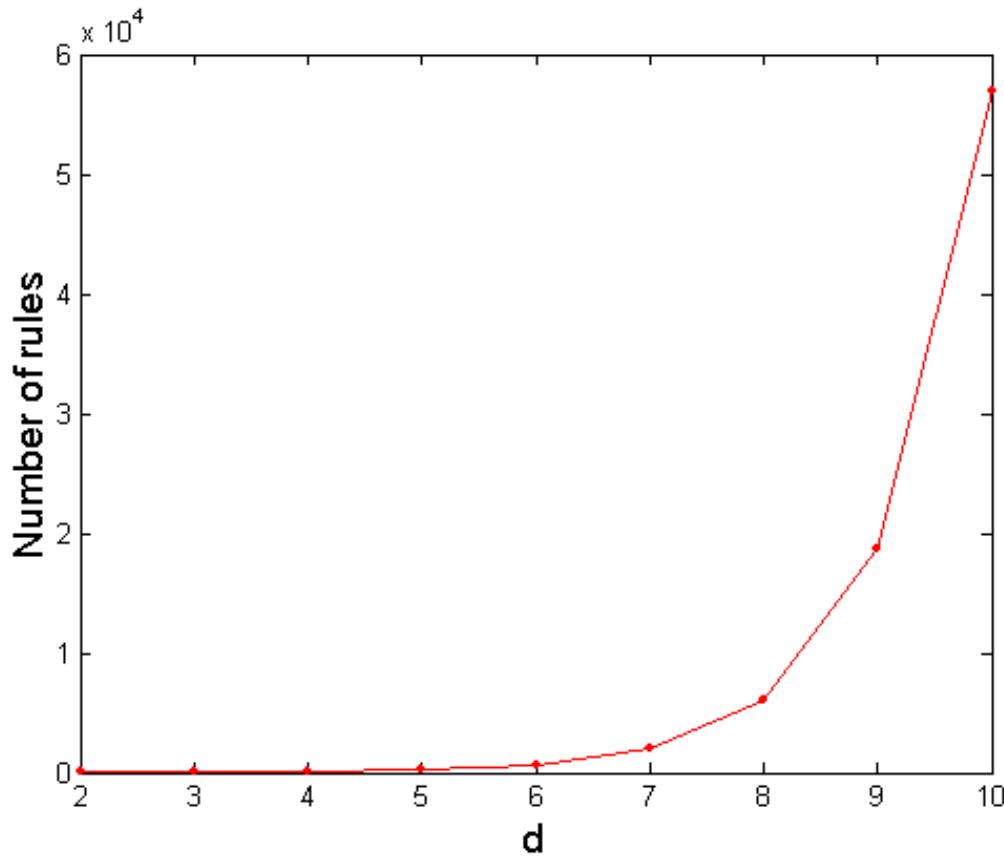


- Match each transaction against every candidate
- Complexity  $\sim O(NMw)$  => **Expensive since  $M = 2^d$  !!!**

# Computational Complexity

□ Given  $d$  unique items:

- Total number of itemsets =  $2^d$
- Total number of possible association rules:



$$\begin{aligned}R &= \sum_{k=1}^{d-1} \binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \\&= 3^d - 2^{d+1} + 1\end{aligned}$$

If  $d=6$ ,  $R = 602$  rules

# Frequent Itemset Generation Strategies

---

## □ Reduce the number of candidates (M)

- Complete search:  $M=2^d$
- Use pruning techniques to reduce M

## □ Reduce the number of transactions (N)

- Reduce size of N as the size of itemset increases
- Used by DHP and vertical-based mining algorithms

## □ Reduce the number of comparisons (NM)

- Use efficient data structures to store the candidates or transactions
- No need to match every candidate against every transaction

# Reducing Number of Candidates

---

## □ Apriori principle:

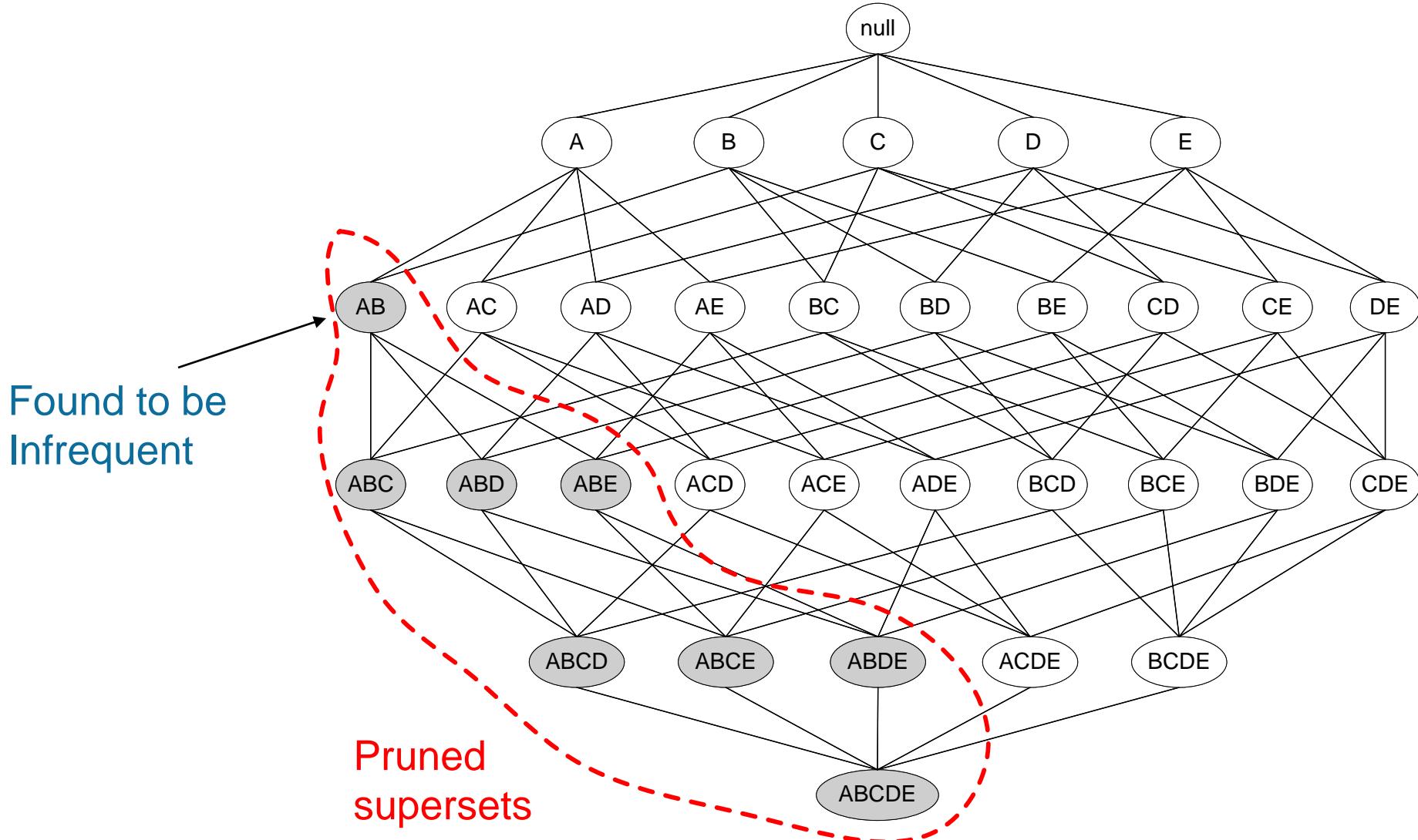
- If an itemset is frequent, then all of its subsets must also be frequent

## □ Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

# Illustrating Apriori Principle



# Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$



Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3



# Apriori Algorithm

---

## □ Method:

- Let  $k=1$
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - ◆ Generate length  $(k+1)$  candidate itemsets from length  $k$  frequent itemsets
  - ◆ Prune candidate itemsets containing subsets of length  $k$  that are infrequent
  - ◆ Count the support of each candidate by scanning the DB
  - ◆ Eliminate candidates that are infrequent, leaving only those that are frequent

# Data Mining Cluster Analysis: Basic Concepts and Algorithms

---

---

## Lecture Notes for Chapter 8

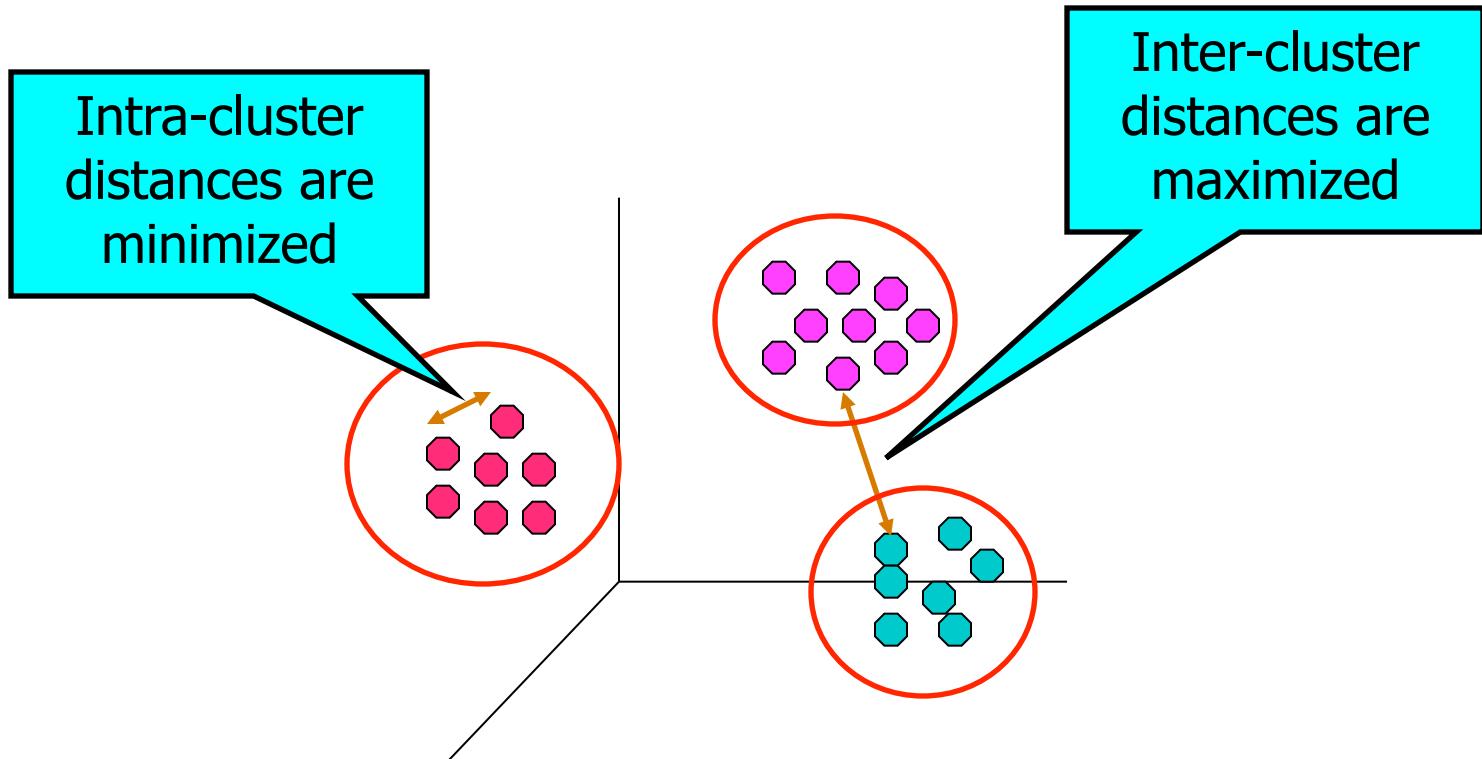
Introduction to Data Mining

by

Tan, Steinbach, Kumar

# What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



# Applications of Cluster Analysis

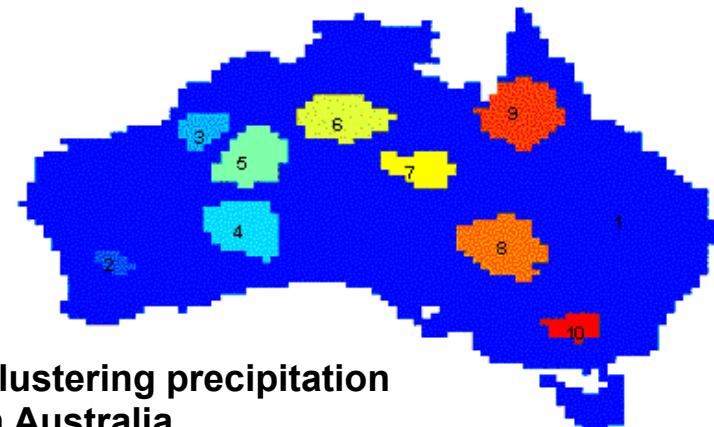
## ● Understanding

- Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations

	<i>Discovered Clusters</i>	<i>Industry Group</i>
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP

## ● Summarization

- Reduce the size of large data sets

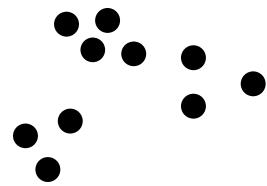


# What is not Cluster Analysis?

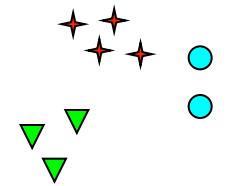
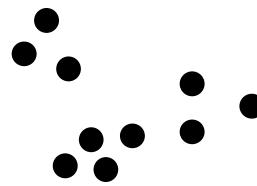
---

- Supervised classification
  - Have class label information
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name
- Results of a query
  - Groupings are a result of an external specification
- Graph partitioning
  - Some mutual relevance and synergy, but areas are not identical

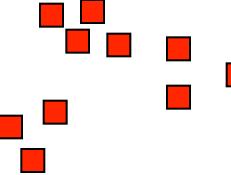
# Notion of a Cluster can be Ambiguous



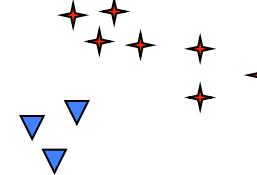
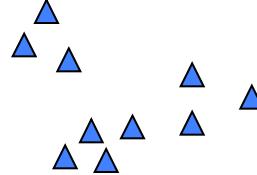
How many clusters?



Six Clusters



Two Clusters



Four Clusters

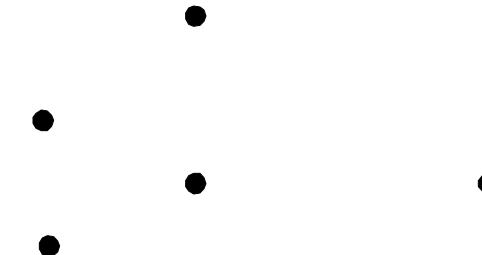
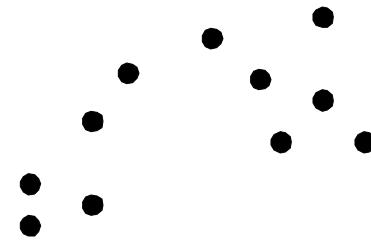
# Types of Clusterings

---

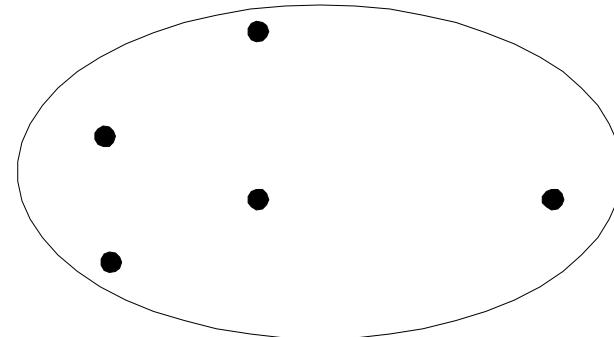
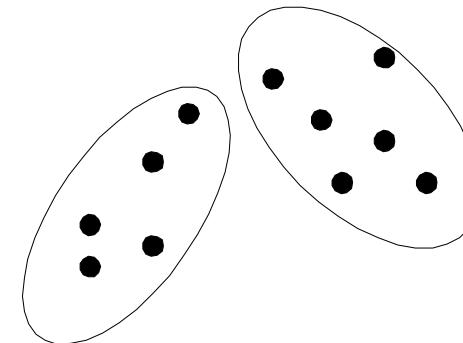
- A **clustering** is a set of clusters
- Important distinction between **hierarchical** and **partitional** sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

# Partitional Clustering

---

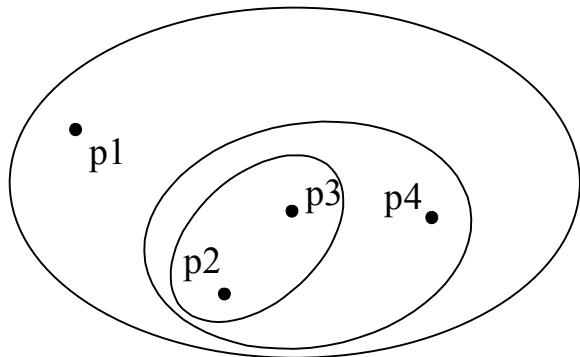


**Original Points**

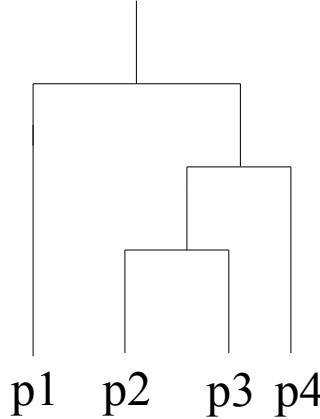


**A Partitional Clustering**

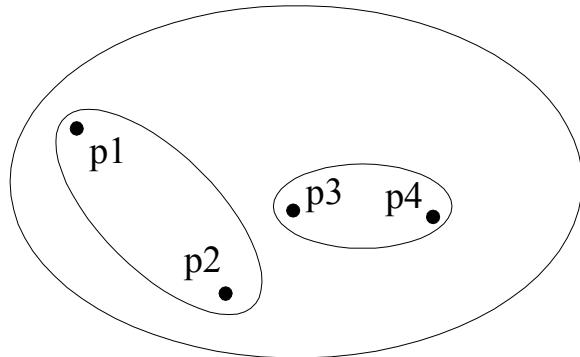
# Hierarchical Clustering



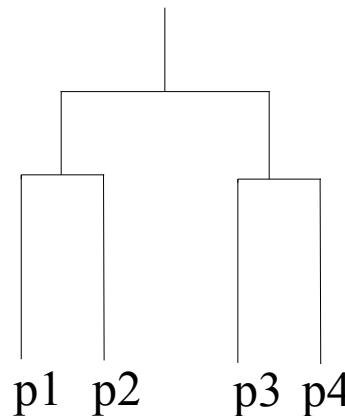
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

# Other Distinctions Between Sets of Clusters

---

- Exclusive versus non-exclusive

- In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or ‘border’ points

- Fuzzy versus non-fuzzy

- In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics

- Partial versus complete

- In some cases, we only want to cluster some of the data

- Heterogeneous versus homogeneous

- Cluster of widely different sizes, shapes, and densities

# Types of Clusters

---

---

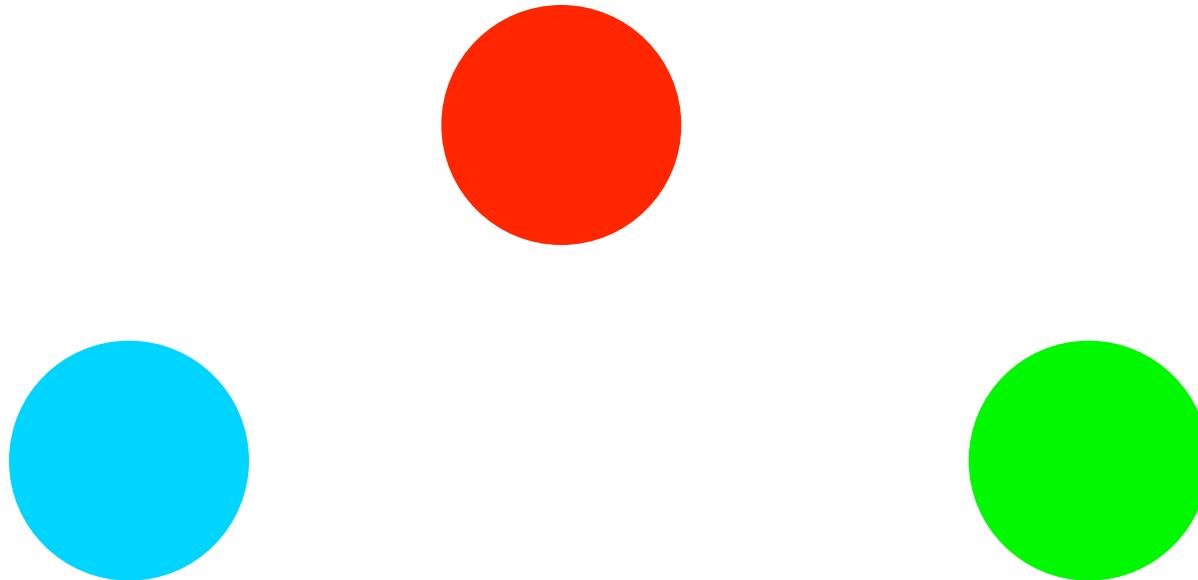
- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

# Types of Clusters: Well-Separated

---

- Well-Separated Clusters:

- A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



**3 well-separated clusters**

# Types of Clusters: Center-Based

---

- Center-based

- A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster
- The center of a cluster is often a **centroid**, the average of all the points in the cluster, or a **medoid**, the most “representative” point of a cluster

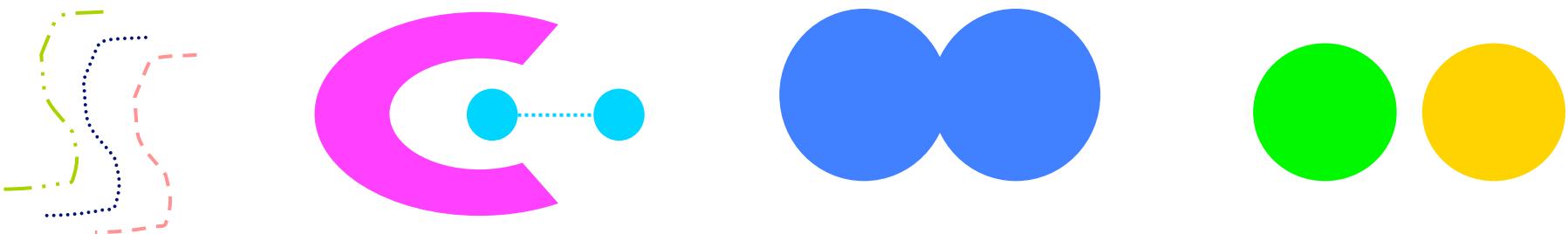


4 center-based clusters

# Types of Clusters: Contiguity-Based

---

- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



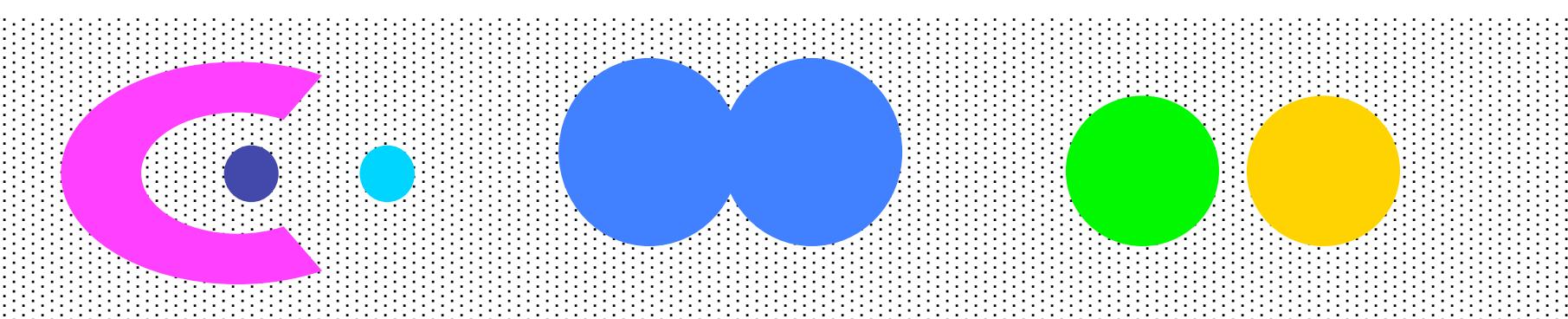
8 contiguous clusters

# Types of Clusters: Density-Based

---

- Density-based

- A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
- Used when the clusters are irregular or intertwined, and when noise and outliers are present.

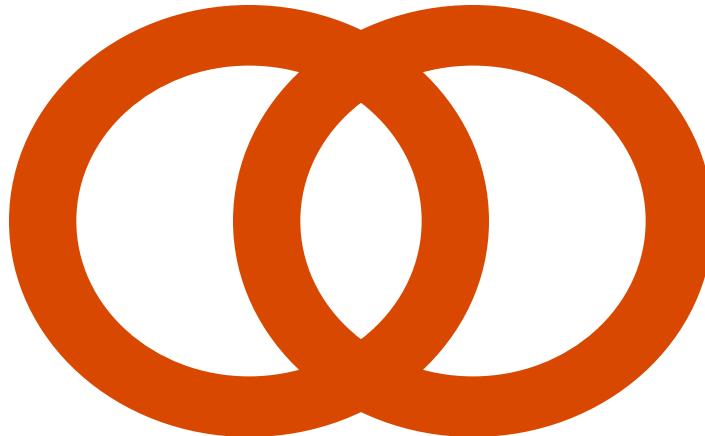


6 density-based clusters

# Types of Clusters: Conceptual Clusters

---

- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.



**2 Overlapping Circles**

# Types of Clusters: Objective Function

---

## ● Clusters Defined by an Objective Function

- Finds clusters that minimize or maximize an objective function.
- Enumerate all possible ways of dividing the points into clusters and evaluate the 'goodness' of each potential set of clusters by using the given objective function. (NP Hard)
- Can have global or local objectives.
  - ◆ Hierarchical clustering algorithms typically have local objectives
  - ◆ Partitional algorithms typically have global objectives
- A variation of the global objective function approach is to fit the data to a parameterized model.
  - ◆ Parameters for the model are determined from the data.
  - ◆ Mixture models assume that the data is a 'mixture' of a number of statistical distributions.

# Types of Clusters: Objective Function ...

---

- Map the clustering problem to a different domain and solve a related problem in that domain
  - Proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points
  - Clustering is equivalent to breaking the graph into connected components, one for each cluster.
  - Want to minimize the edge weight between clusters and maximize the edge weight within clusters

# Characteristics of the Input Data Are Important

---

- Type of proximity or density measure
  - This is a derived measure, but central to clustering
- Sparseness
  - Dictates type of similarity
  - Adds to efficiency
- Attribute type
  - Dictates type of similarity
- Type of Data
  - Dictates type of similarity
  - Other characteristics, e.g., autocorrelation
- Dimensionality
- Noise and Outliers
- Type of Distribution

# Clustering Algorithms

---

---

- K-means and its variants
- Hierarchical clustering
- Density-based clustering

# K-means Clustering

---

- Partitional clustering approach
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters,  $K$ , must be specified
- The basic algorithm is very simple

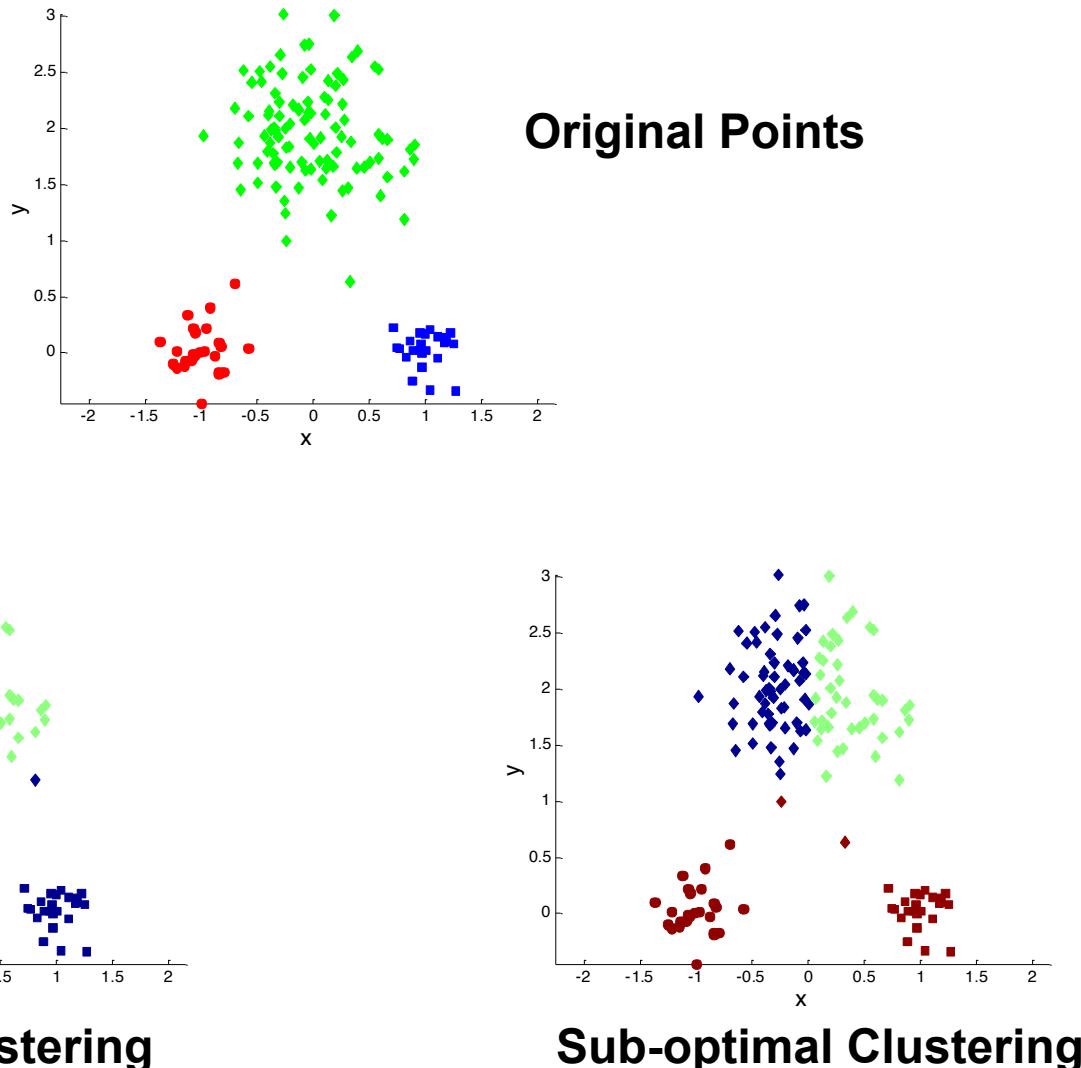
- 
- 1: Select  $K$  points as the initial centroids.
  - 2: **repeat**
  - 3:     Form  $K$  clusters by assigning all points to the closest centroid.
  - 4:     Recompute the centroid of each cluster.
  - 5: **until** The centroids don't change
-

# K-means Clustering – Details

---

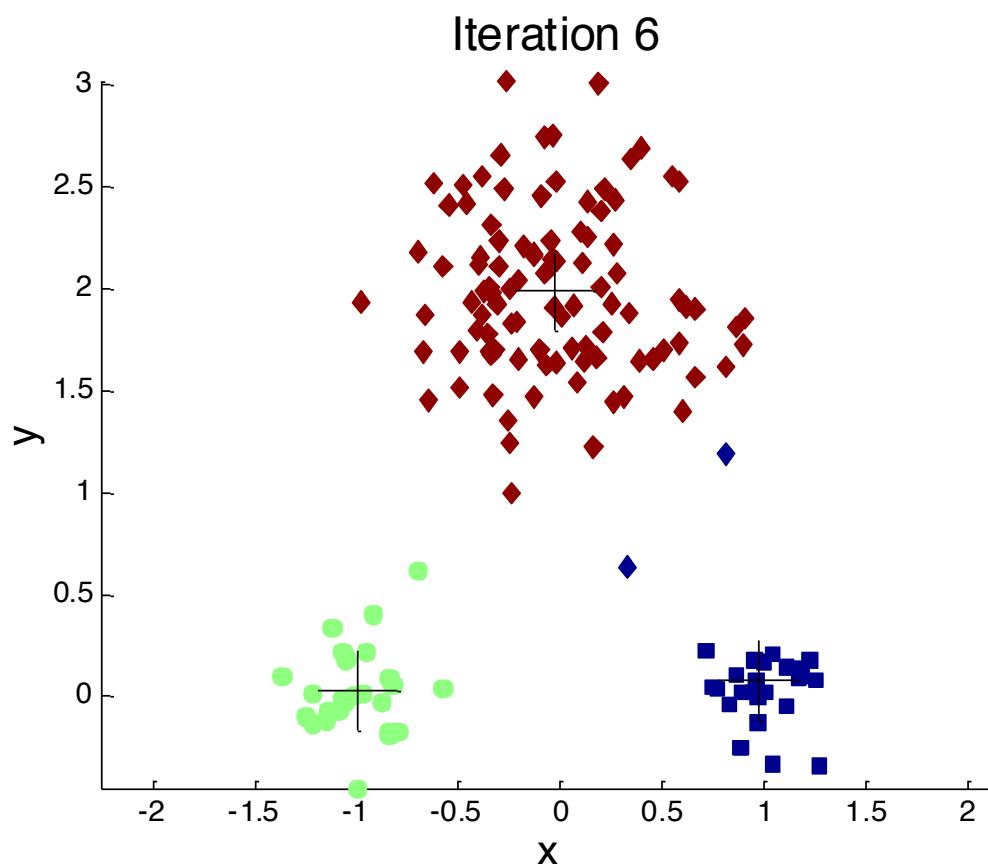
- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is  $O( n * K * I * d )$ 
  - $n$  = number of points,  $K$  = number of clusters,  
 $I$  = number of iterations,  $d$  = number of attributes

# Two different K-means Clusterings

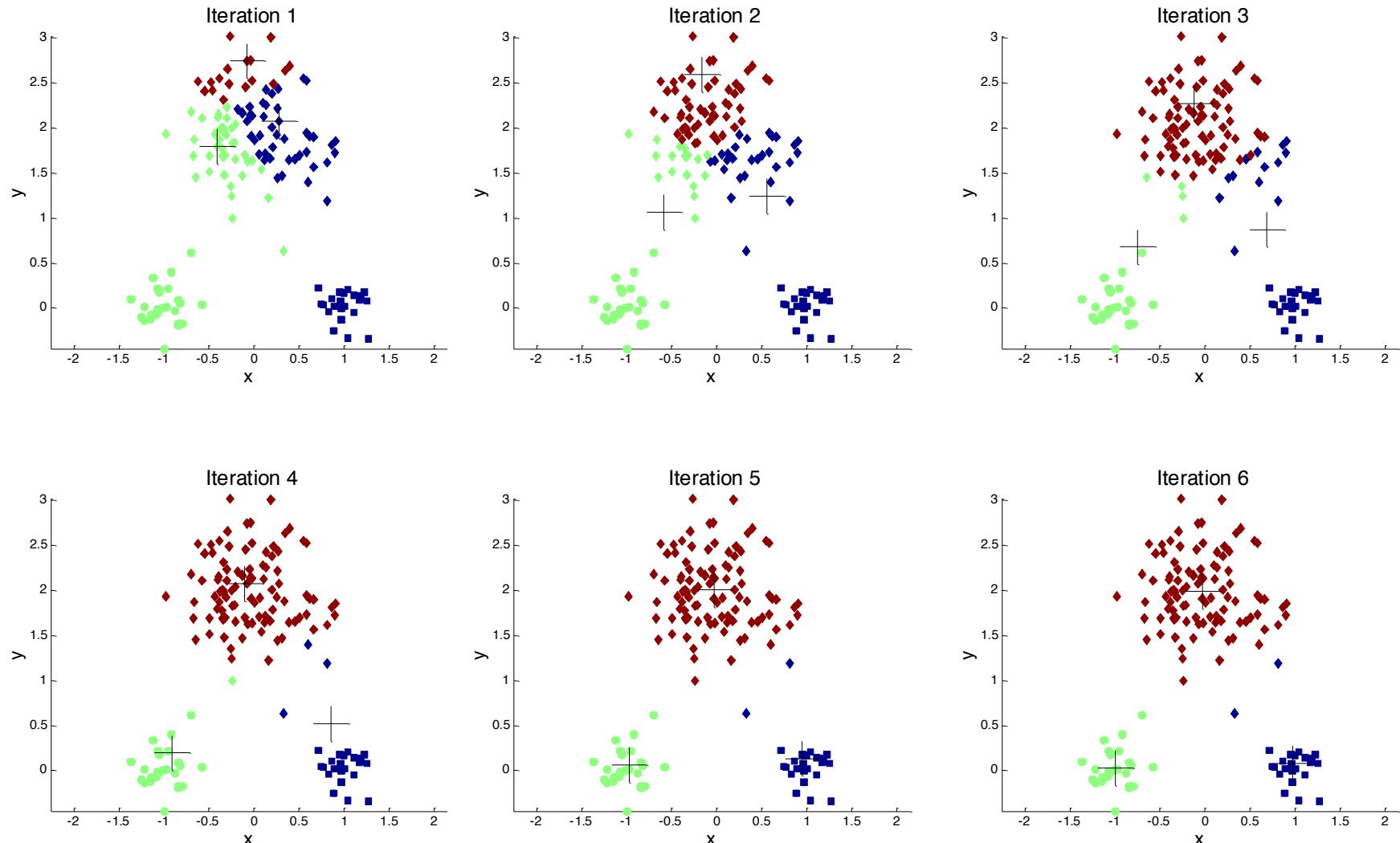


# Importance of Choosing Initial Centroids

---



# Importance of Choosing Initial Centroids



# Evaluating K-means Clusters

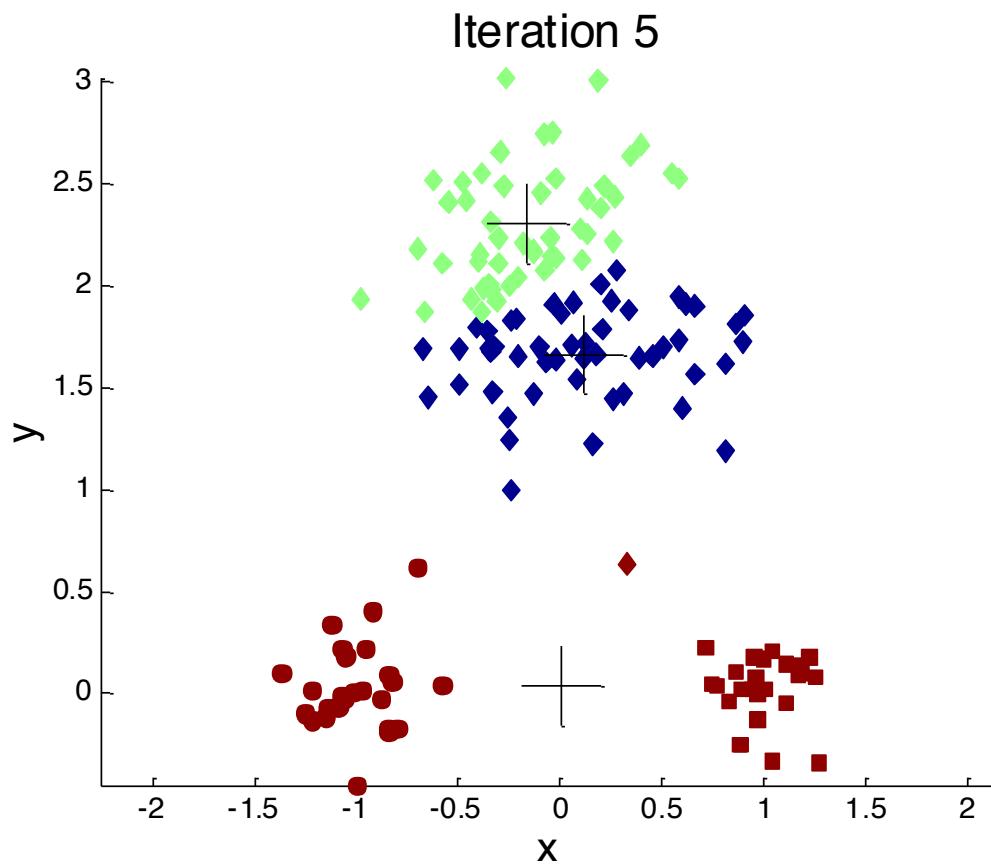
---

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

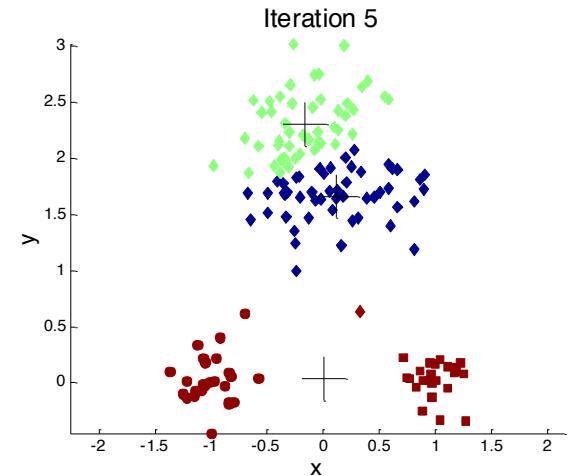
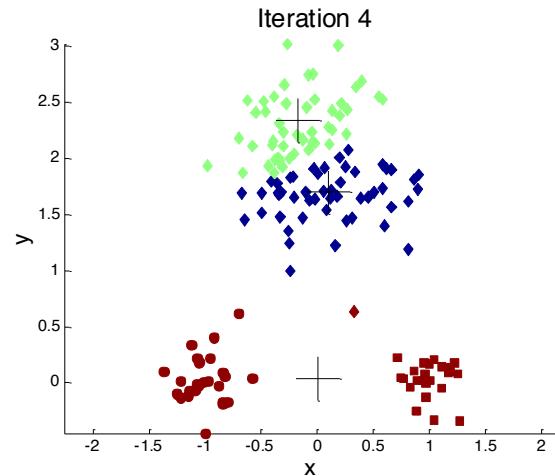
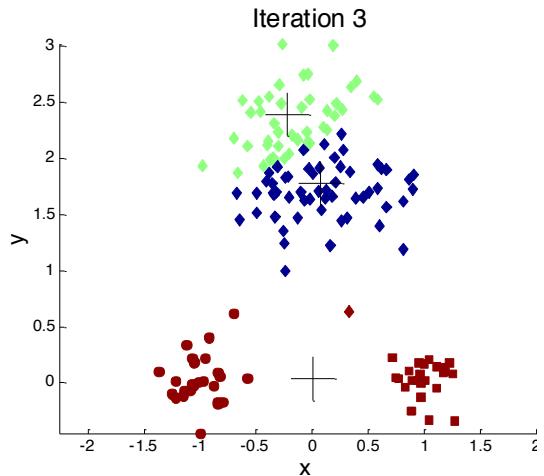
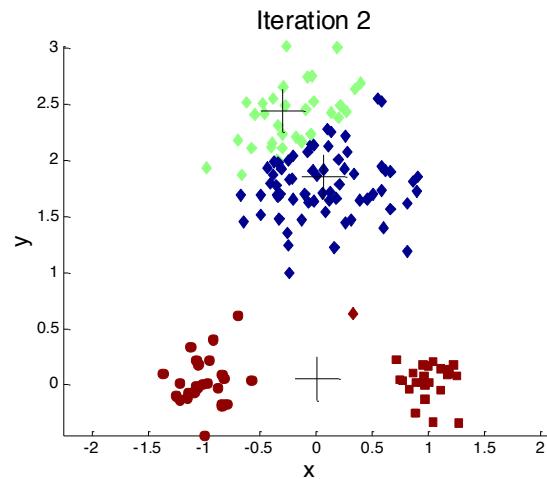
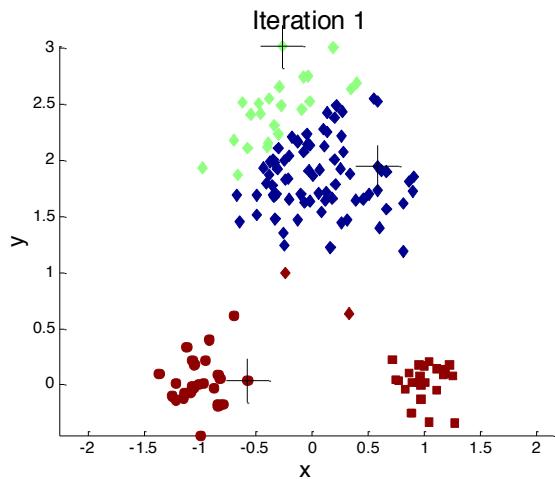
$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- $x$  is a data point in cluster  $C_i$  and  $m_i$  is the representative point for cluster  $C_i$ 
  - ◆ can show that  $m_i$  corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase  $K$ , the number of clusters
  - ◆ A good clustering with smaller  $K$  can have a lower SSE than a poor clustering with higher  $K$

# Importance of Choosing Initial Centroids ...



# Importance of Choosing Initial Centroids ...



# Problems with Selecting Initial Points

---

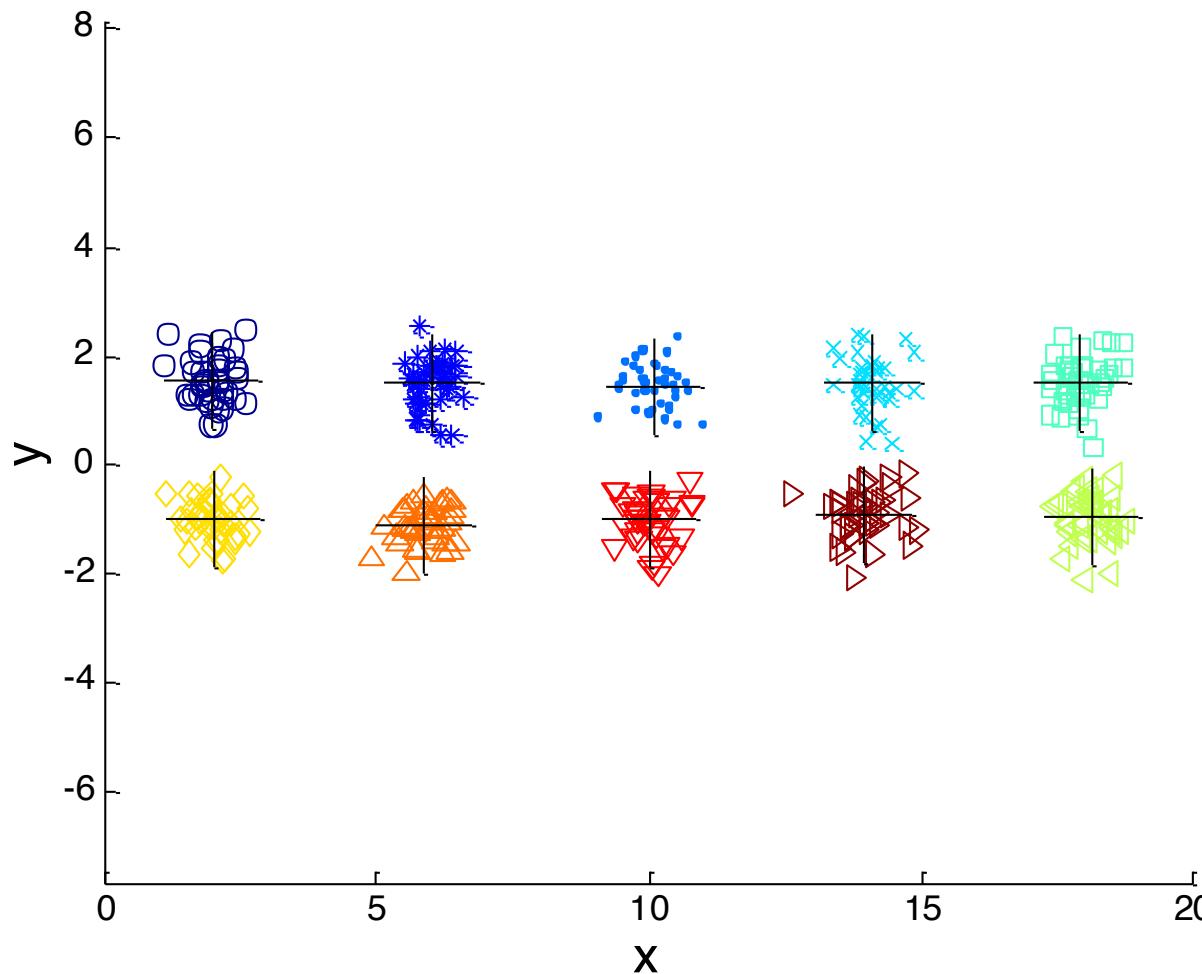
- If there are  $K$  ‘real’ clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when  $K$  is large
  - If clusters are the same size,  $n$ , then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if  $K = 10$ , then probability =  $10!/10^{10} = 0.00036$
- Sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t
- Consider an example of five pairs of clusters

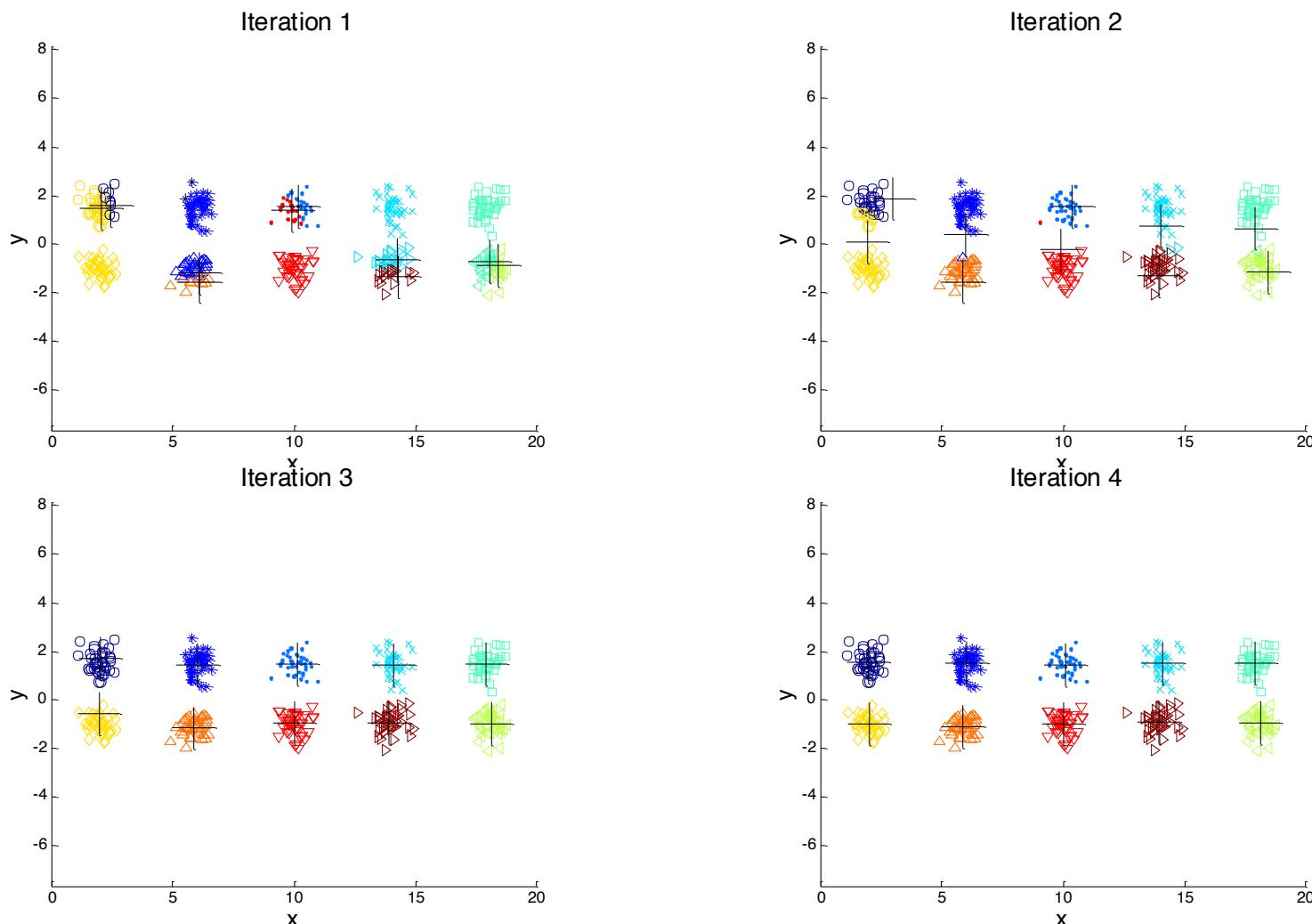
# 10 Clusters Example

Iteration 4



Starting with two initial centroids in one cluster of each pair of clusters

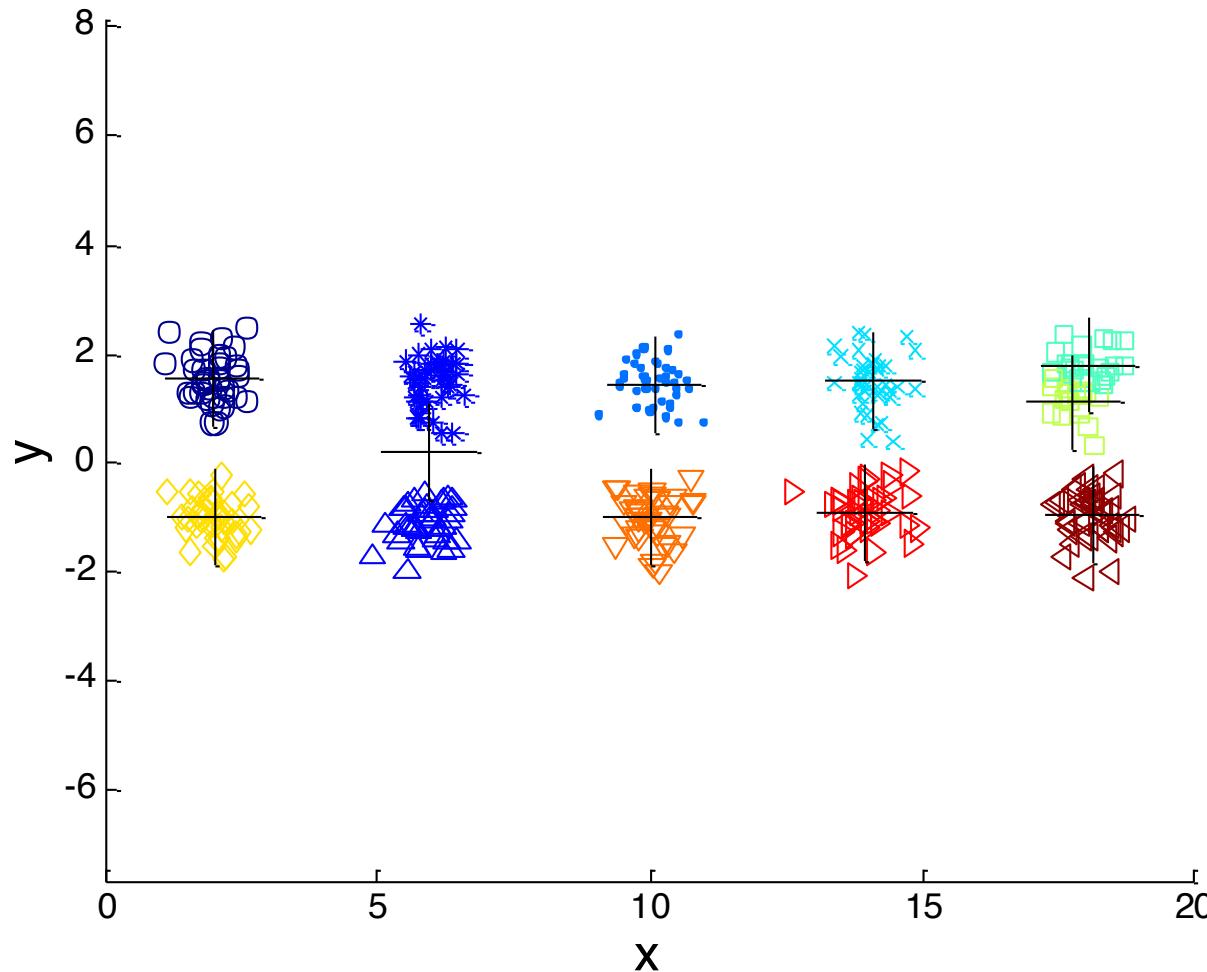
# 10 Clusters Example



**Starting with two initial centroids in one cluster of each pair of clusters**

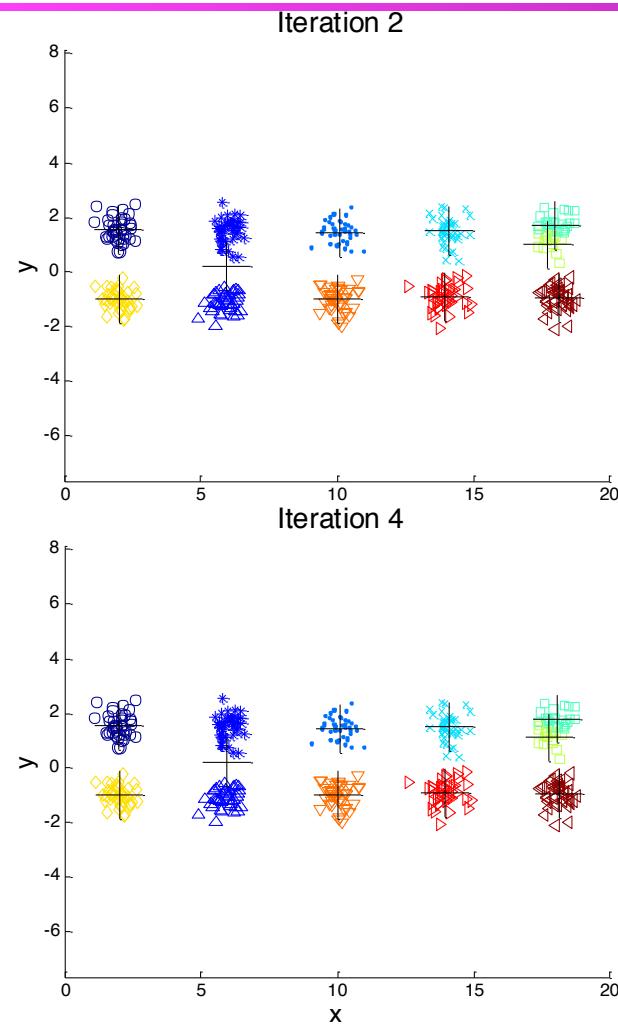
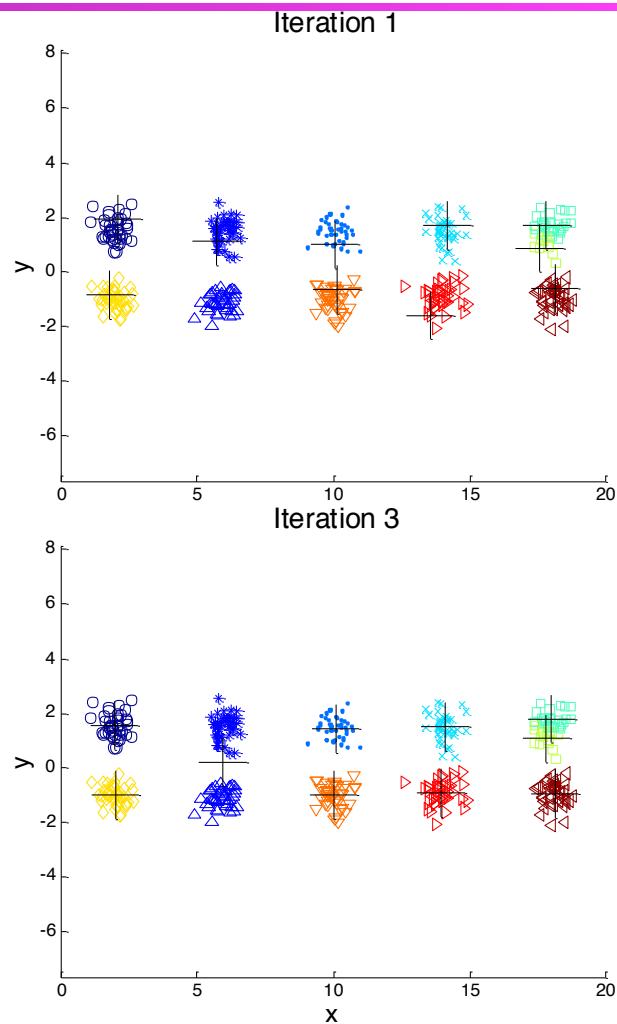
# 10 Clusters Example

Iteration 4



Starting with some pairs of clusters having three initial centroids, while other have only one.

# 10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

# Solutions to Initial Centroids Problem

---

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than  $k$  initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing
- Bisecting K-means
  - Not as susceptible to initialization issues

# Pre-processing and Post-processing

---

## ● Pre-processing

- Normalize the data
- Eliminate outliers

## ● Post-processing

- Eliminate small clusters that may represent outliers
- Split ‘loose’ clusters, i.e., clusters with relatively high SSE
- Merge clusters that are ‘close’ and that have relatively low SSE
- Can use these steps during the clustering process

# Bisecting K-means

---

- Bisecting K-means algorithm

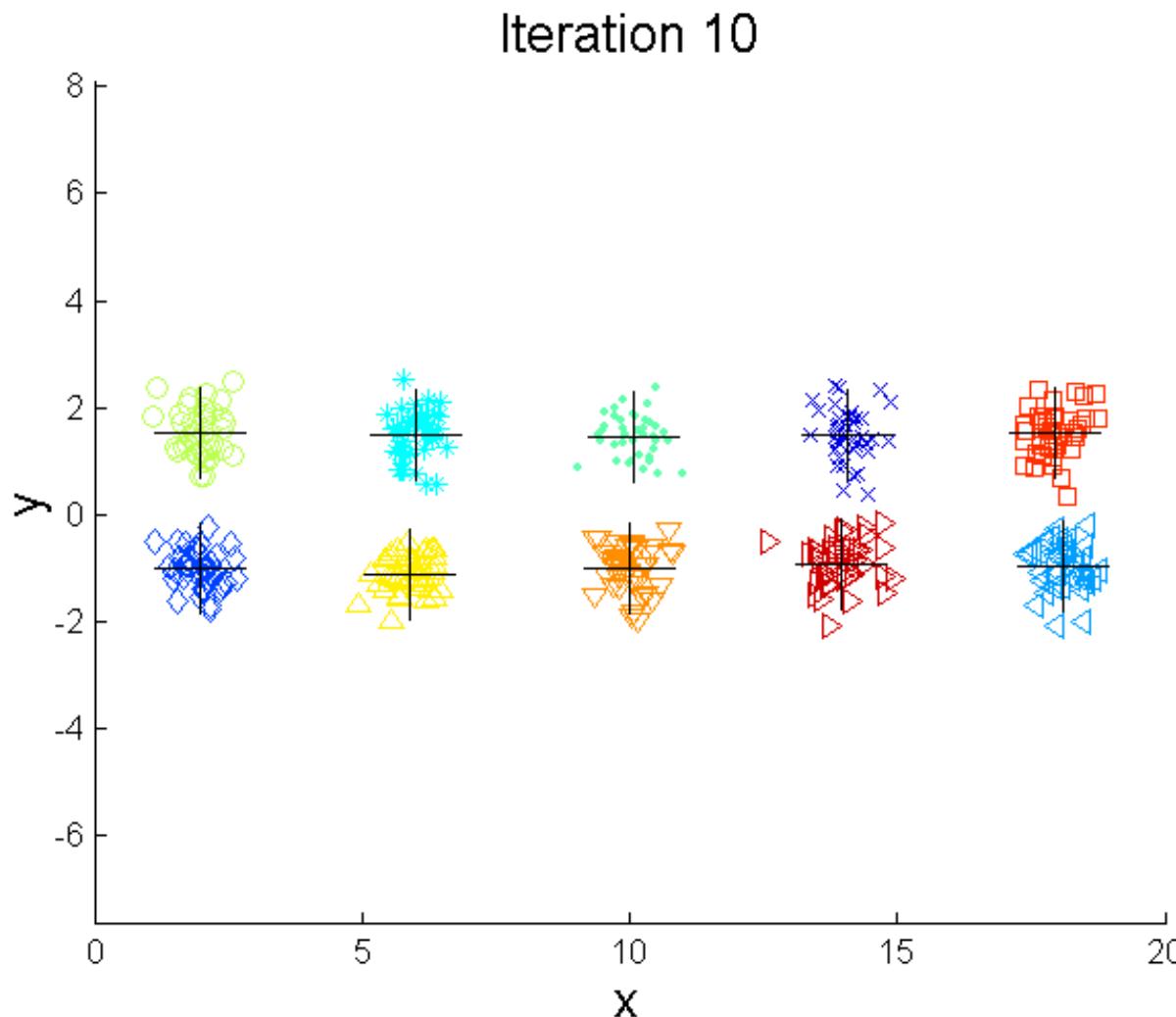
- Variant of K-means that can produce a partitional or a hierarchical clustering

---

```
1: Initialize the list of clusters to contain the cluster containing all points.  
2: repeat  
3:   Select a cluster from the list of clusters  
4:   for  $i = 1$  to number_of_iterations do  
5:     Bisect the selected cluster using basic K-means  
6:   end for  
7:   Add the two clusters from the bisection with the lowest SSE to the list of clusters.  
8: until Until the list of clusters contains  $K$  clusters
```

---

# Bisecting K-means Example

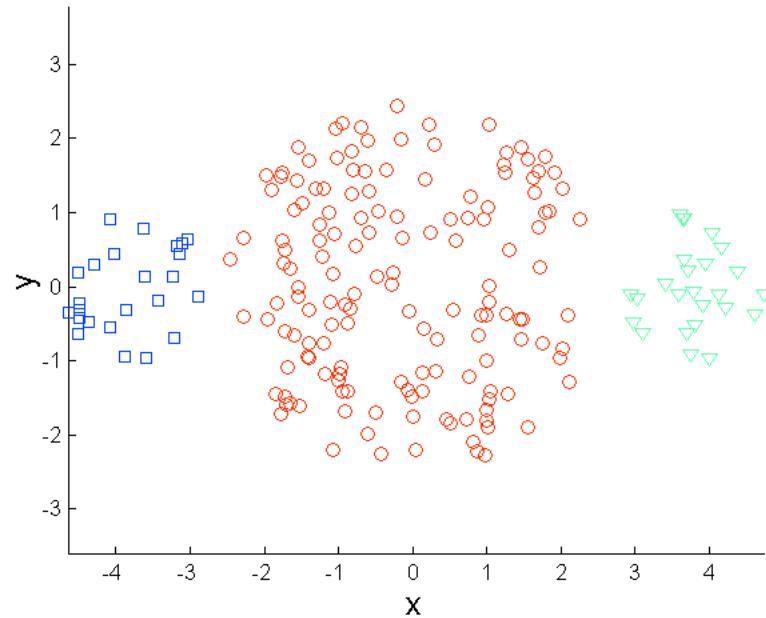


# Limitations of K-means

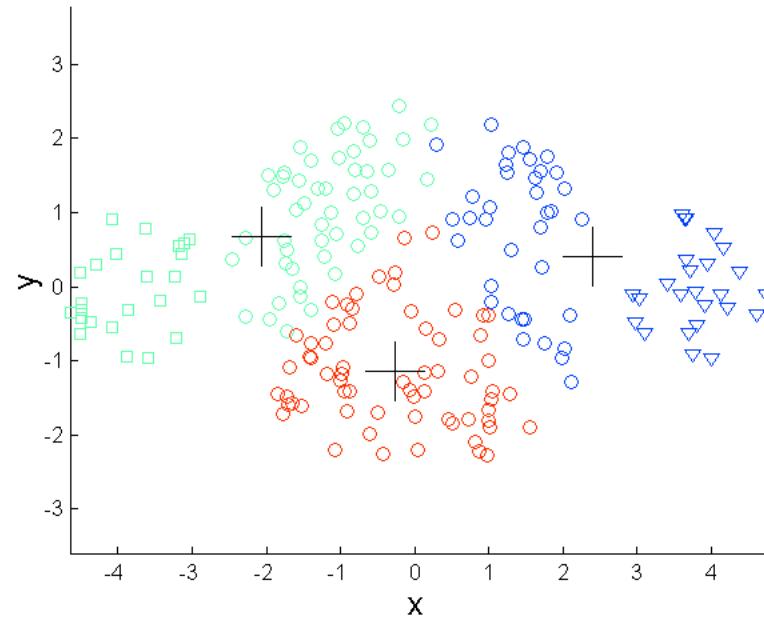
---

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

# Limitations of K-means: Differing Sizes

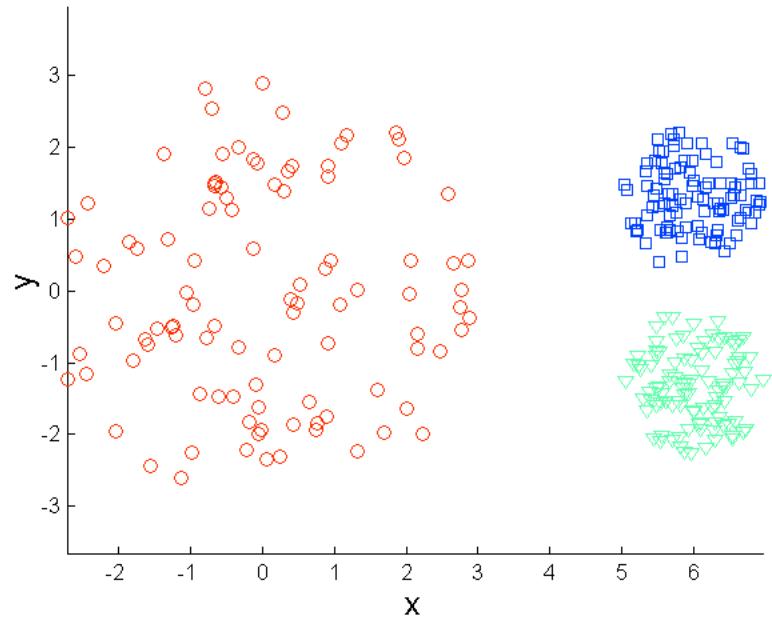


Original Points

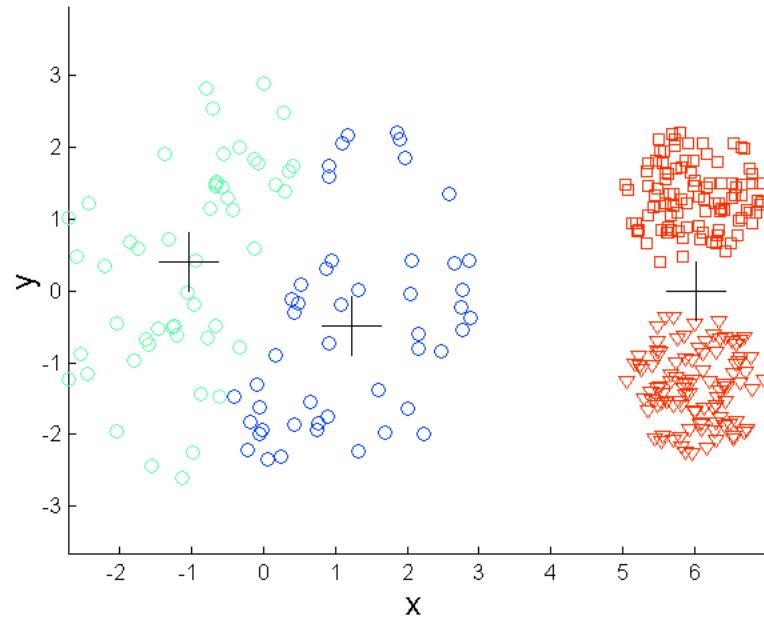


K-means (3 Clusters)

# Limitations of K-means: Differing Density

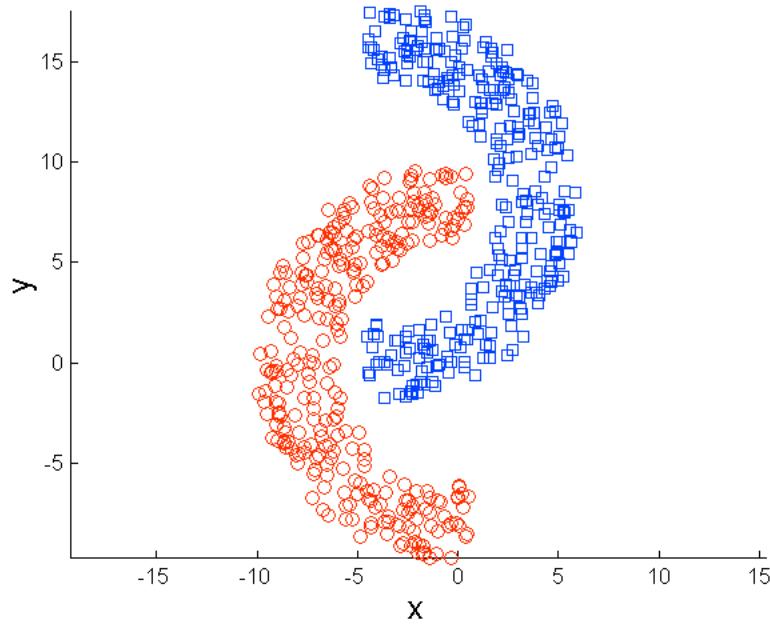


Original Points

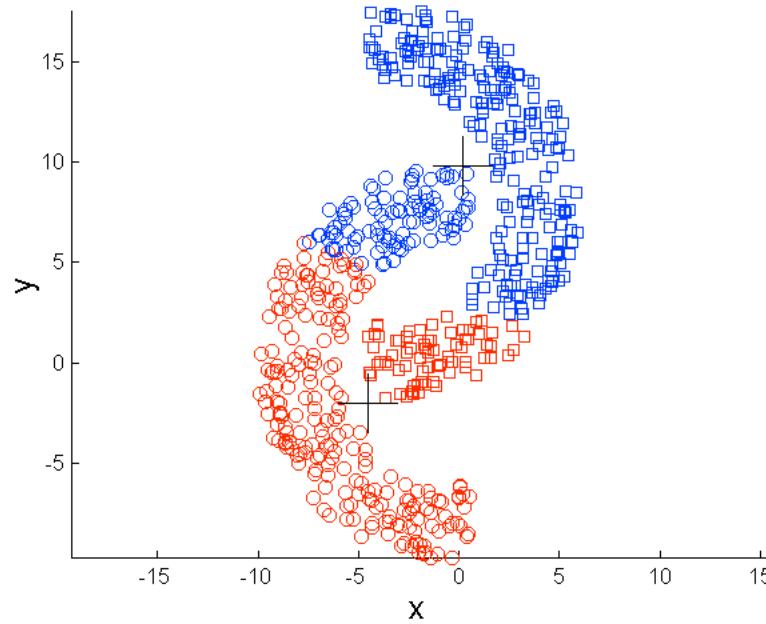


K-means (3 Clusters)

# Limitations of K-means: Non-globular Shapes

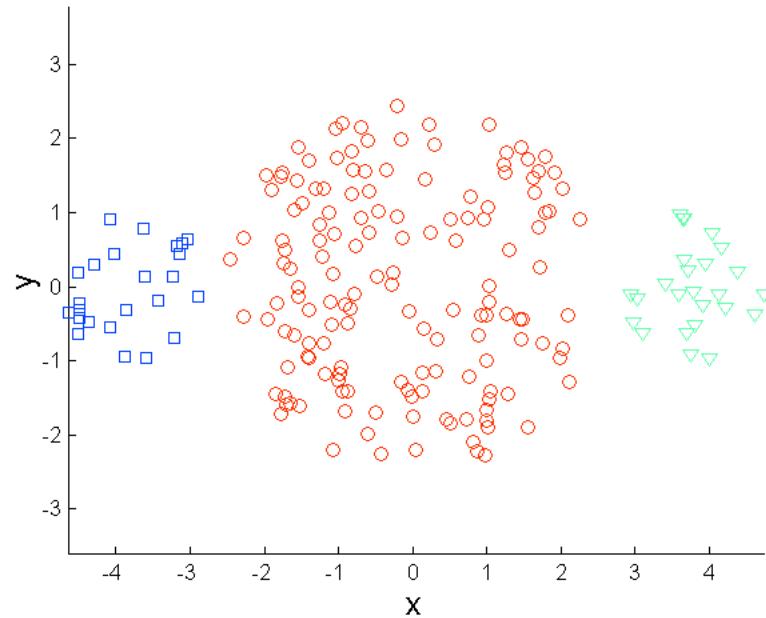


Original Points

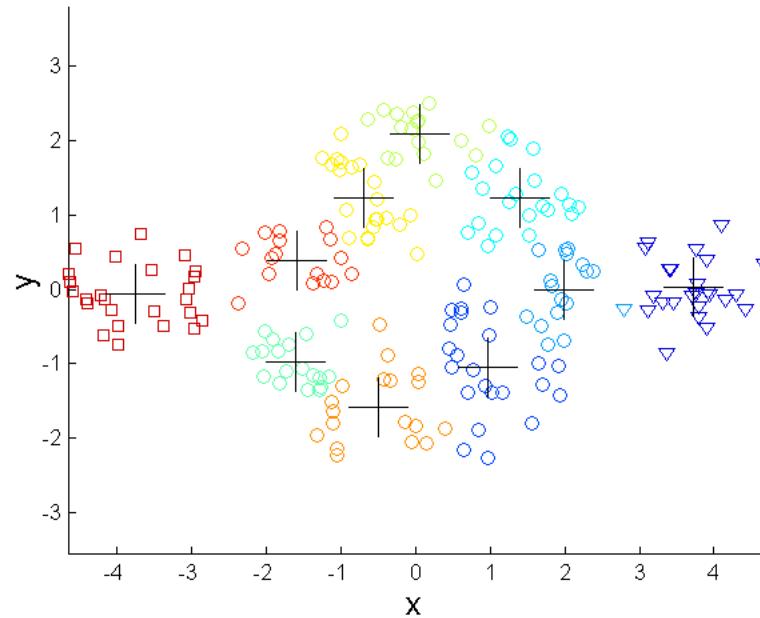


K-means (2 Clusters)

# Overcoming K-means Limitations



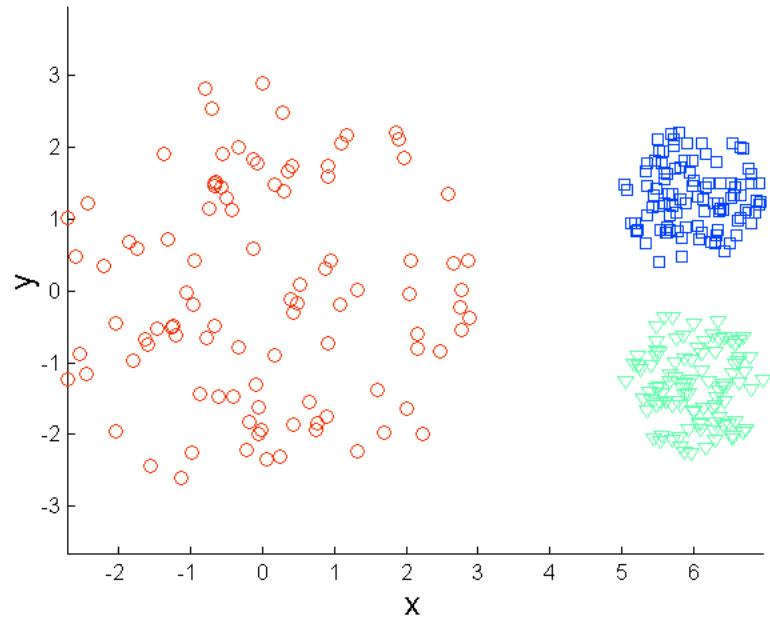
**Original Points**



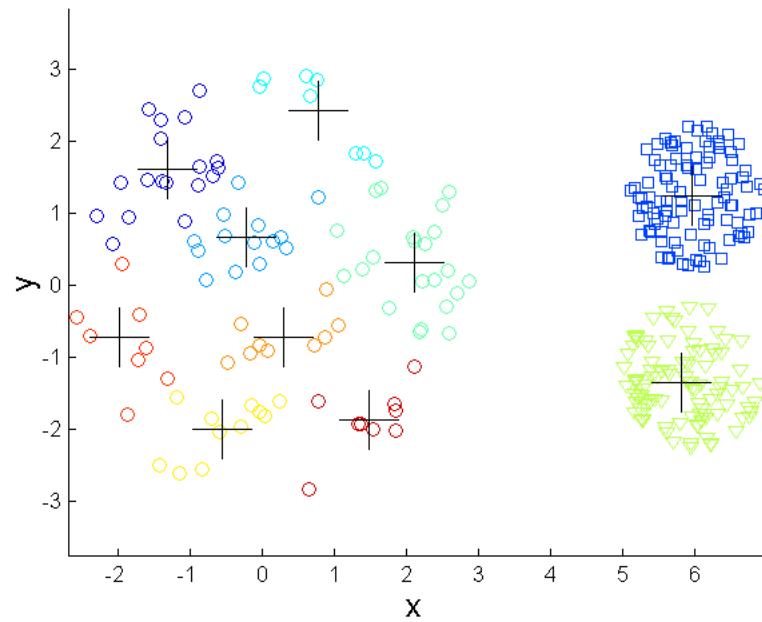
**K-means Clusters**

One solution is to use many clusters.  
Find parts of clusters, but need to put together.

# Overcoming K-means Limitations

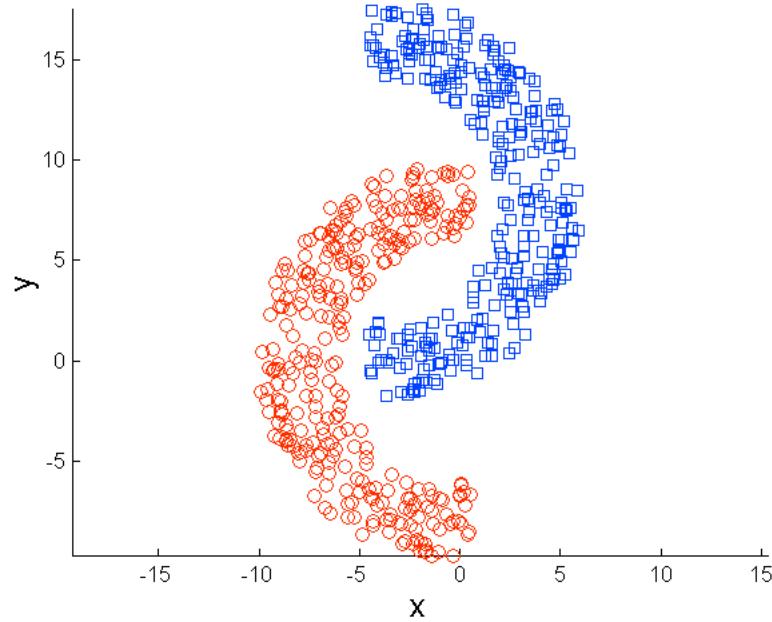


Original Points

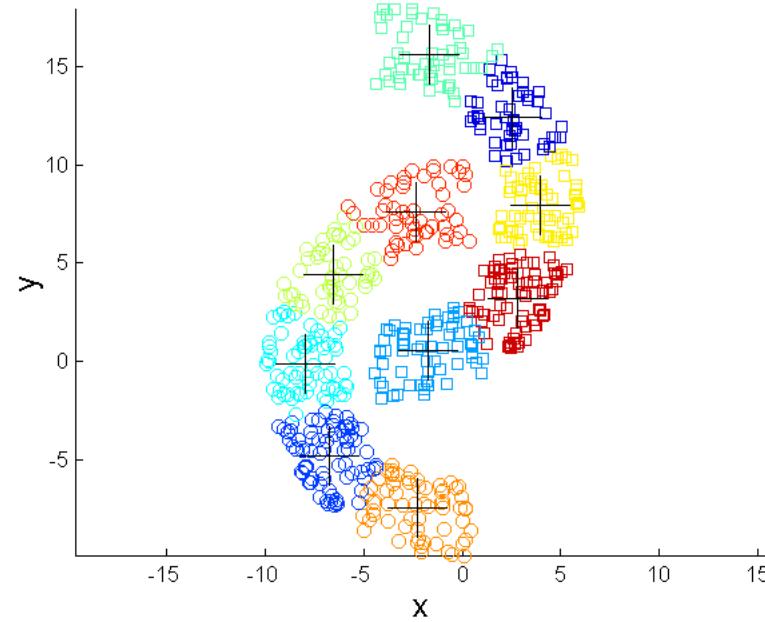


K-means Clusters

# Overcoming K-means Limitations



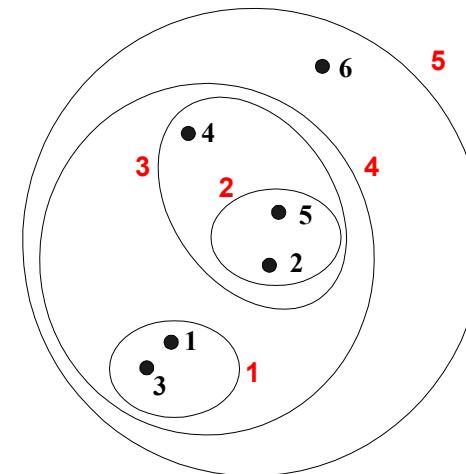
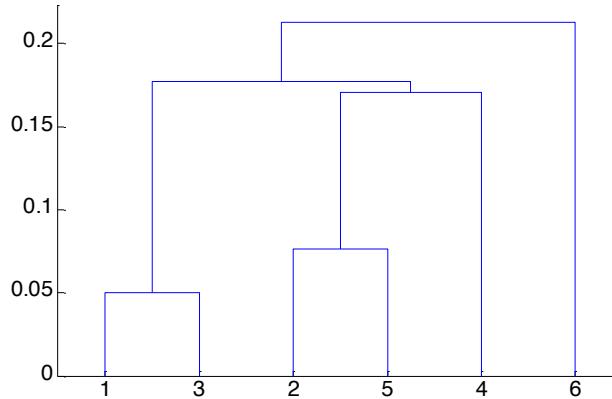
Original Points



K-means Clusters

# Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



# Strengths of Hierarchical Clustering

---

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)

# Hierarchical Clustering

---

- Two main types of hierarchical clustering
  - Agglomerative:
    - ◆ Start with the points as individual clusters
    - ◆ At each step, merge the closest pair of clusters until only one cluster (or  $k$  clusters) left
  - Divisive:
    - ◆ Start with one, all-inclusive cluster
    - ◆ At each step, split a cluster until each cluster contains a point (or there are  $k$  clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

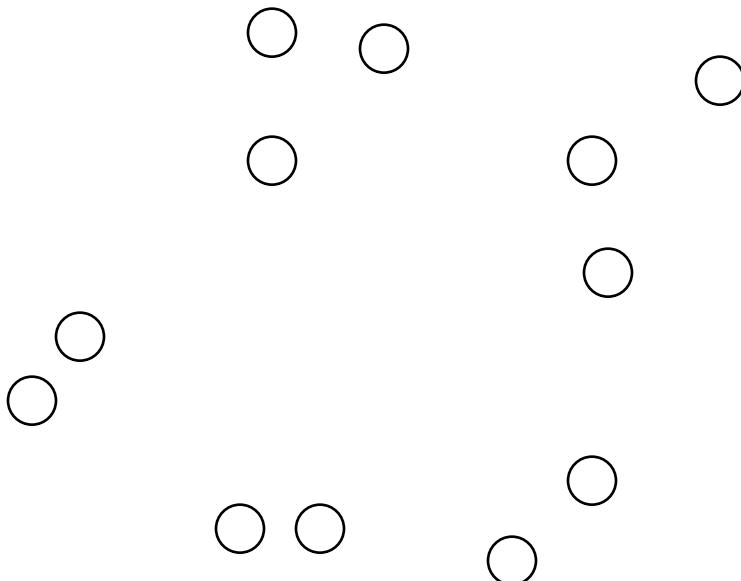
# Agglomerative Clustering Algorithm

---

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

# Starting Situation

- Start with clusters of individual points and a proximity matrix



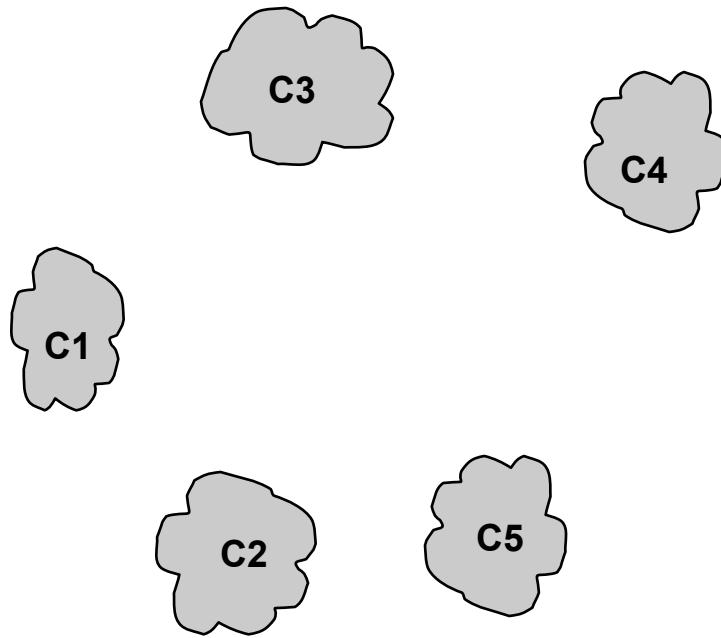
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

**Proximity Matrix**



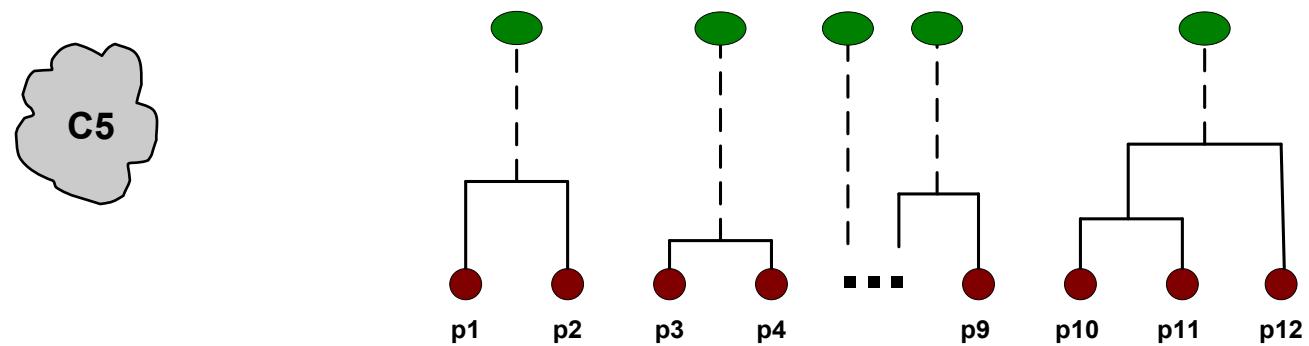
# Intermediate Situation

- After some merging steps, we have some clusters



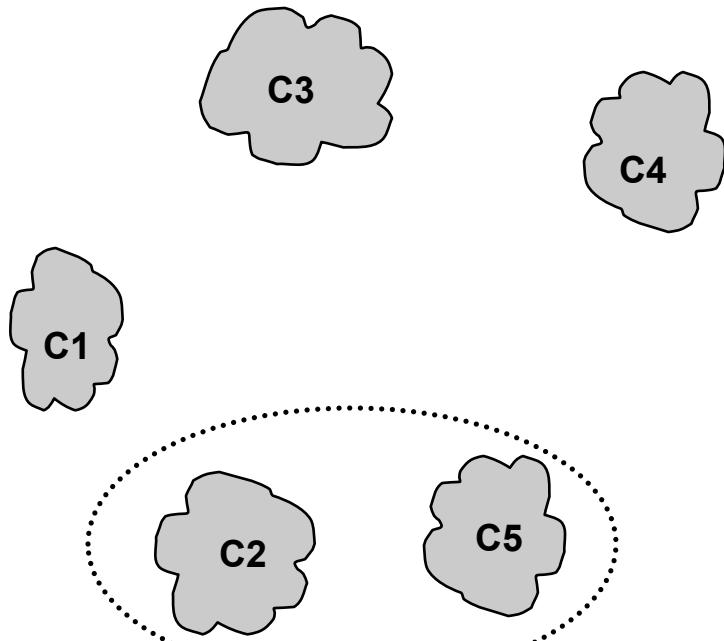
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



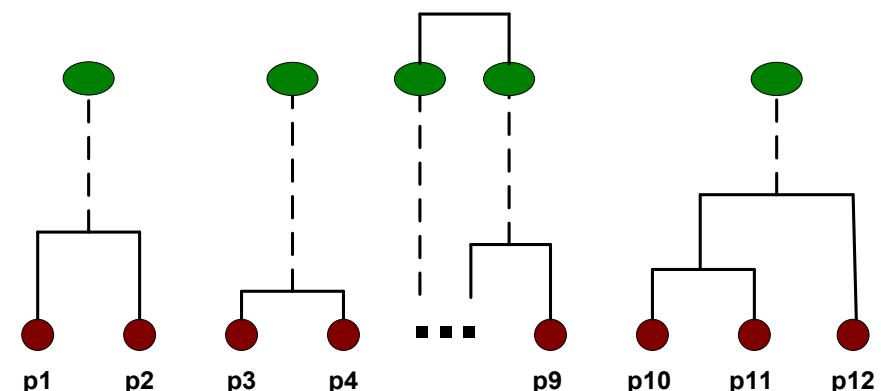
# Intermediate Situation

- We want to merge the two closest clusters ( $C_2$  and  $C_5$ ) and update the proximity matrix.



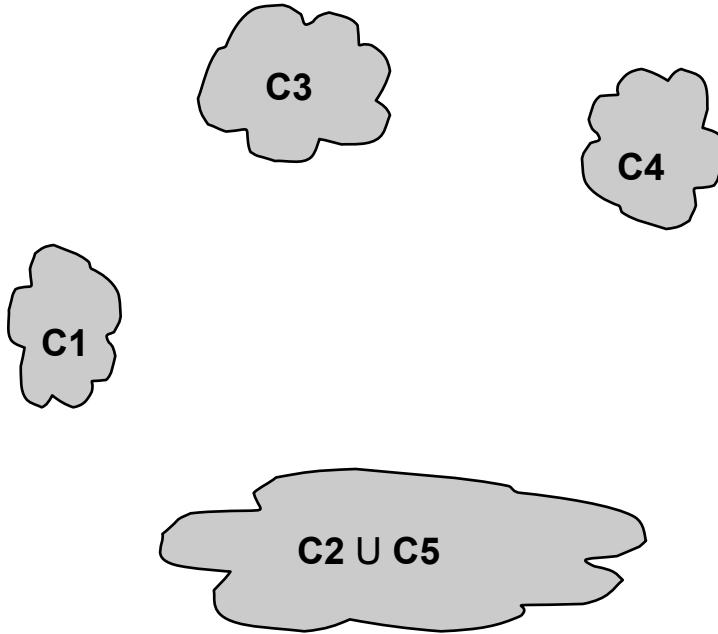
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



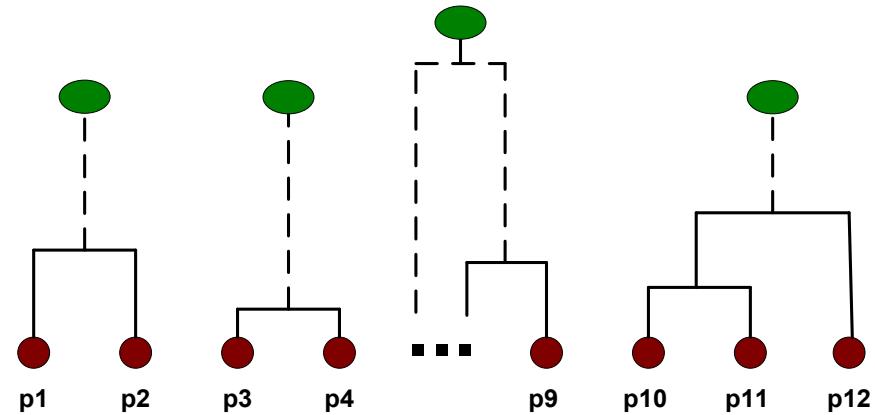
# After Merging

- The question is “How do we update the proximity matrix?”

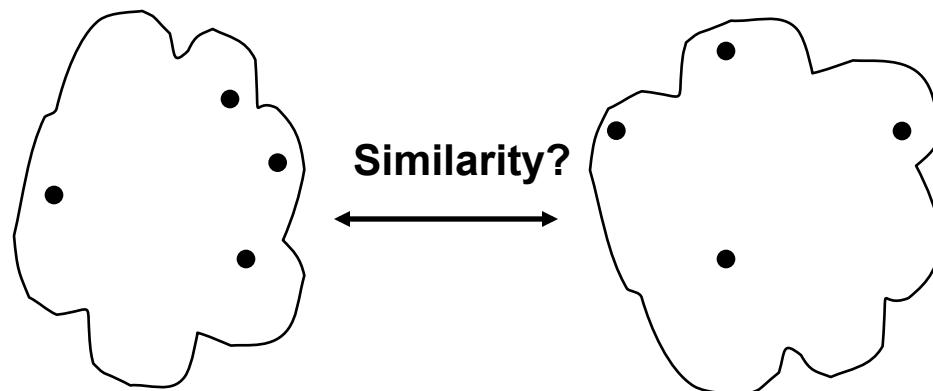


		C2 U C1	C5	C3	C4
C1	C1	?			
	C2 U C5	?	?	?	?
C3		?			
C4		?			

Proximity Matrix



# How to Define Inter-Cluster Similarity

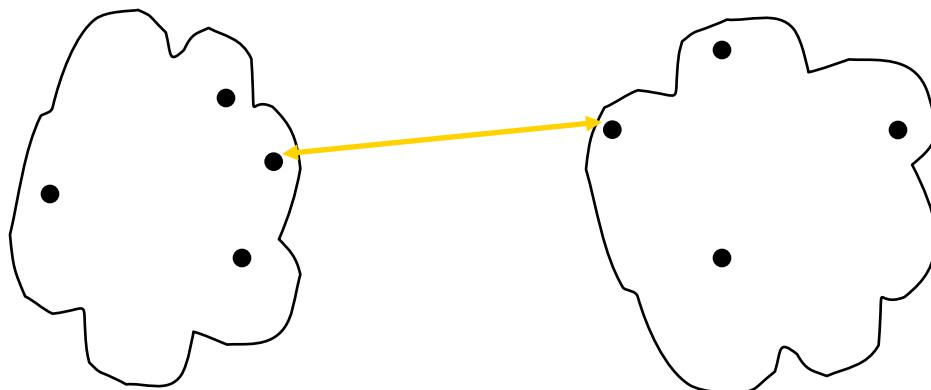


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.	.	.	.	.	.	.

● **Proximity Matrix**

# How to Define Inter-Cluster Similarity

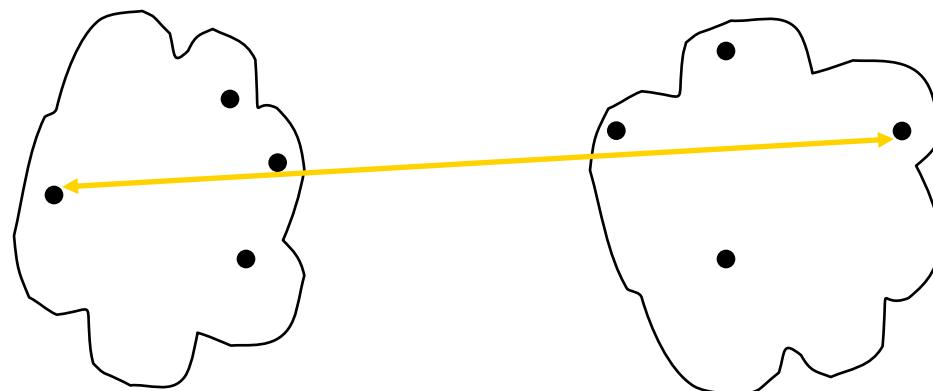


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

# How to Define Inter-Cluster Similarity

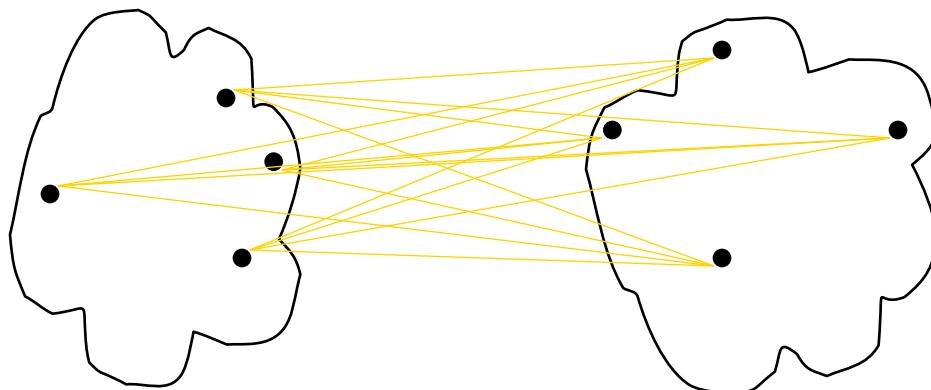


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.	.	.	.	.	.	.

Proximity Matrix

# How to Define Inter-Cluster Similarity

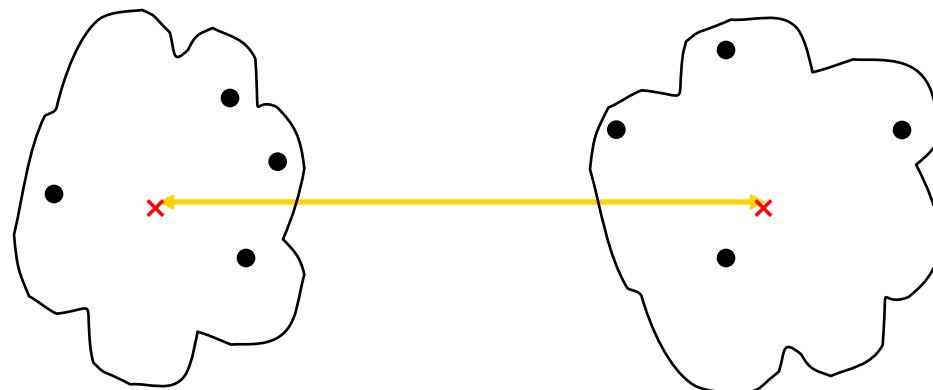


- MIN
- MAX
- **Group Average**
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**

# How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

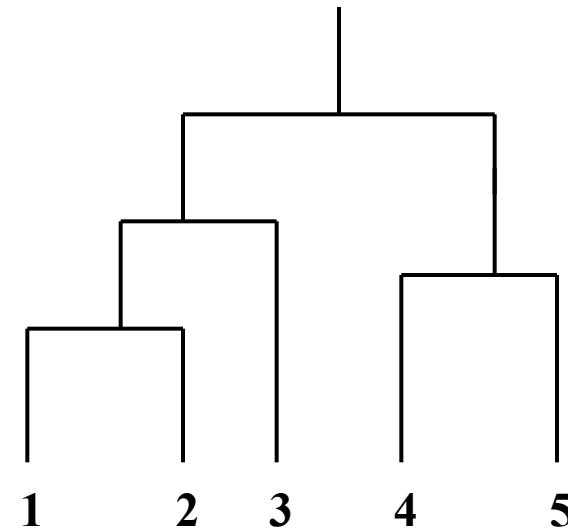
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

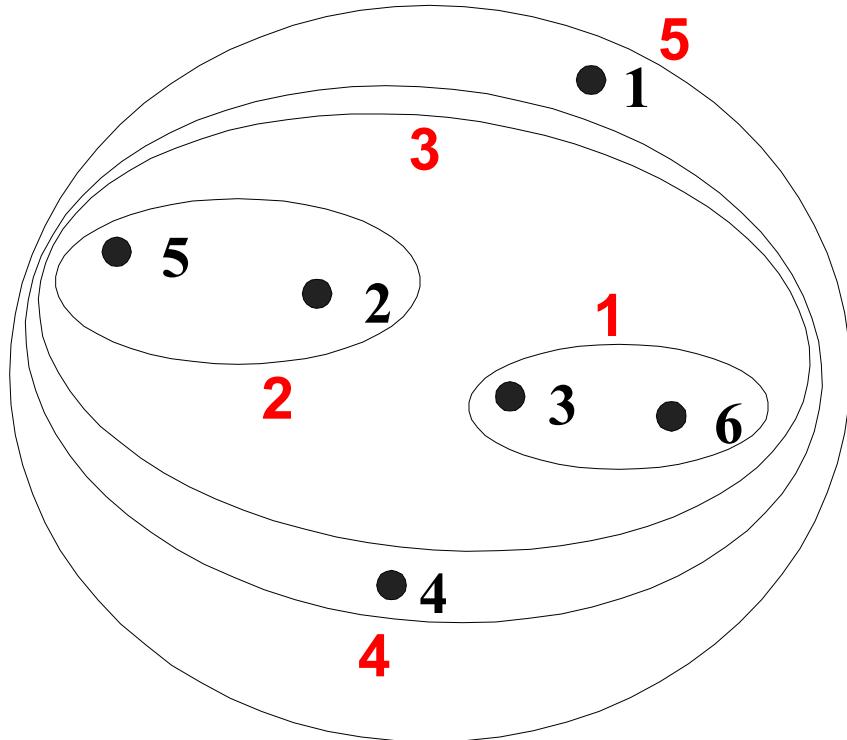
# Cluster Similarity: MIN or Single Link

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

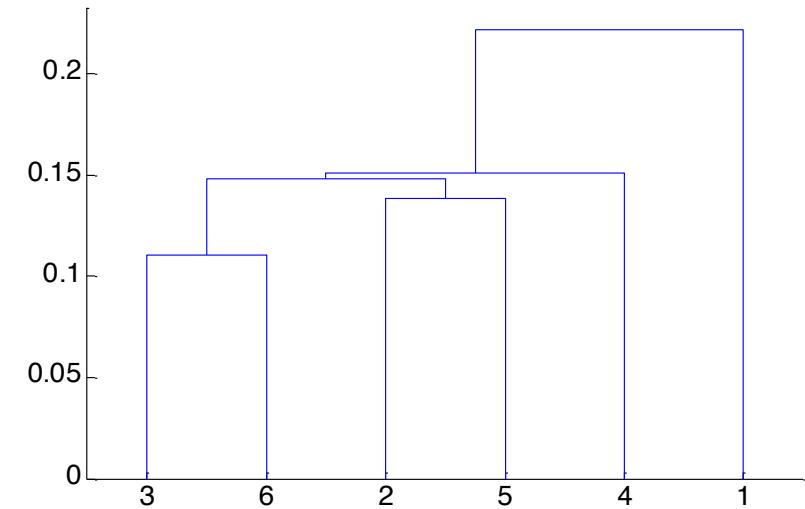
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: MIN



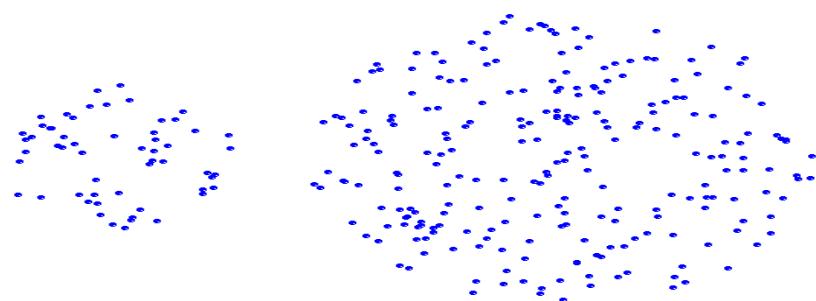
Nested Clusters



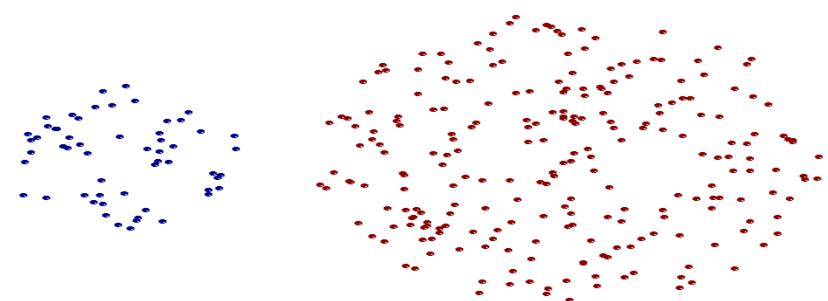
Dendrogram

# Strength of MIN

---



Original Points

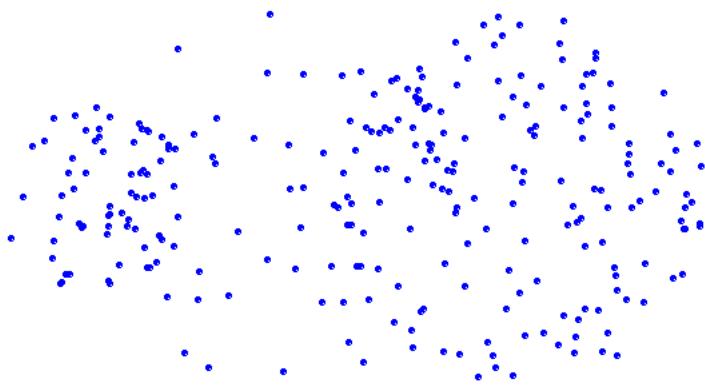


Two Clusters

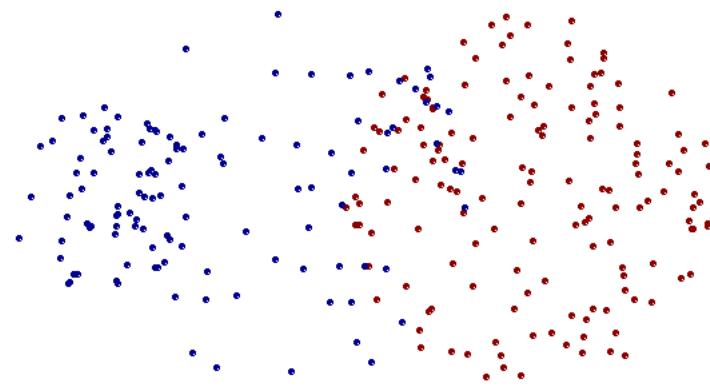
- Can handle non-elliptical shapes

# Limitations of MIN

---



Original Points



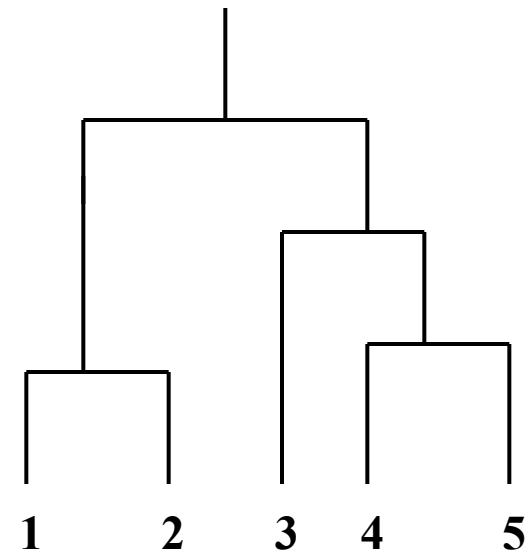
Two Clusters

- Sensitive to noise and outliers

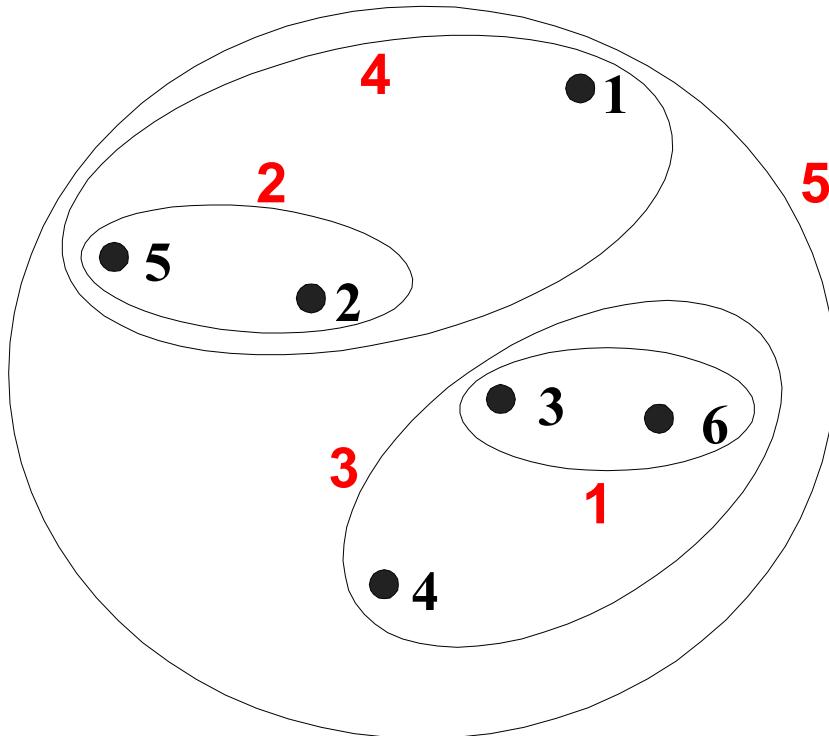
# Cluster Similarity: MAX or Complete Linkage

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

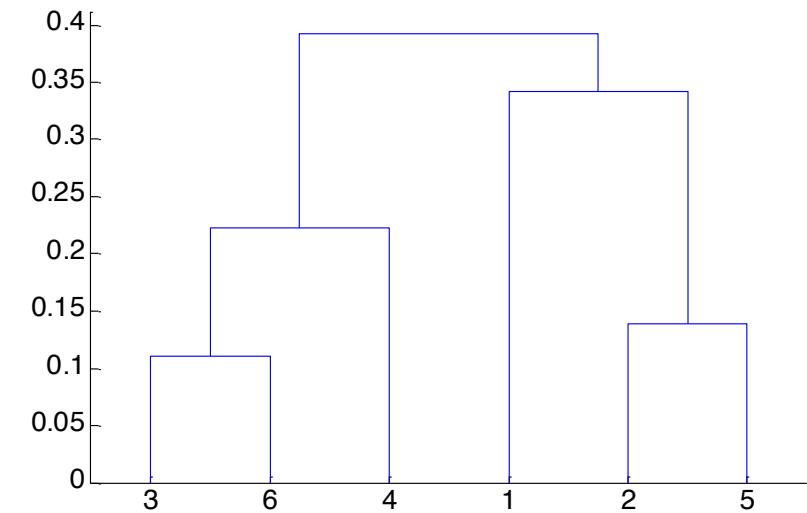
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: MAX



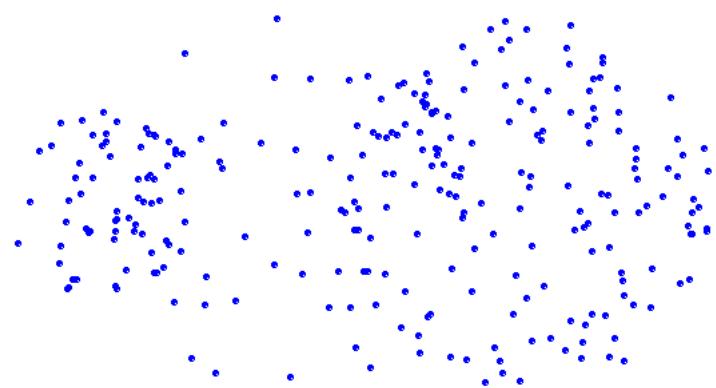
Nested Clusters



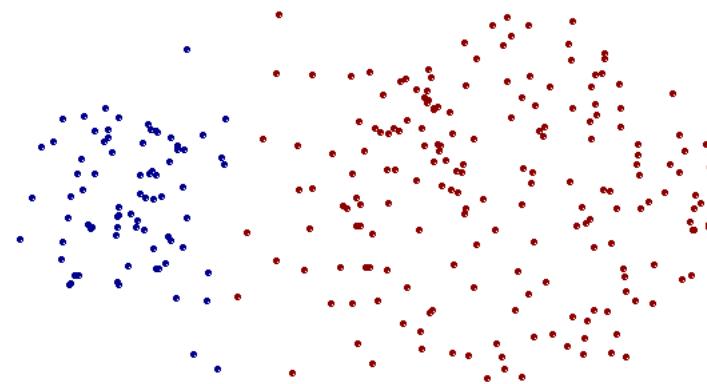
Dendrogram

# Strength of MAX

---



Original Points

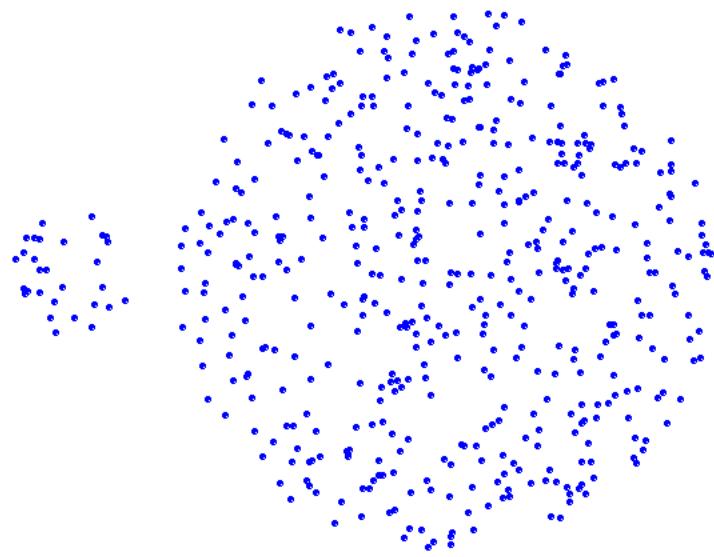


Two Clusters

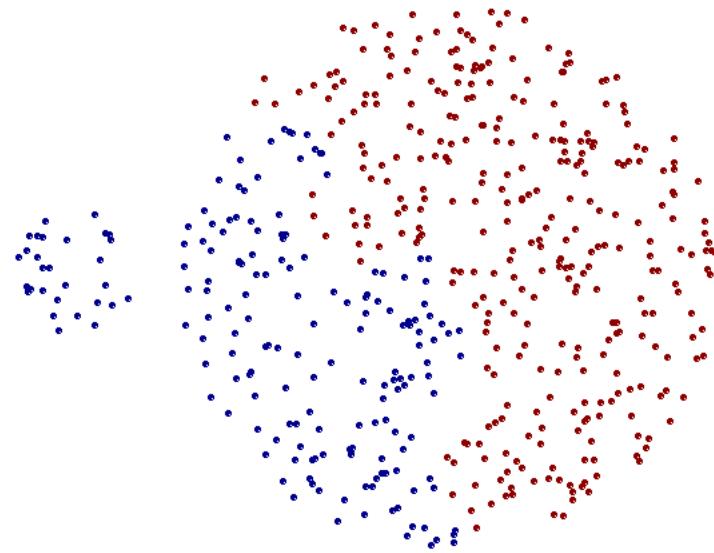
- Less susceptible to noise and outliers

# Limitations of MAX

---



**Original Points**



**Two Clusters**

- Tends to break large clusters
- Biased towards globular clusters

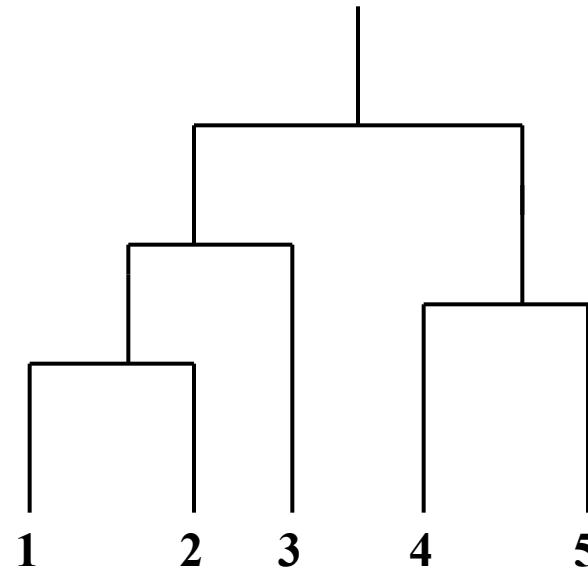
# Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

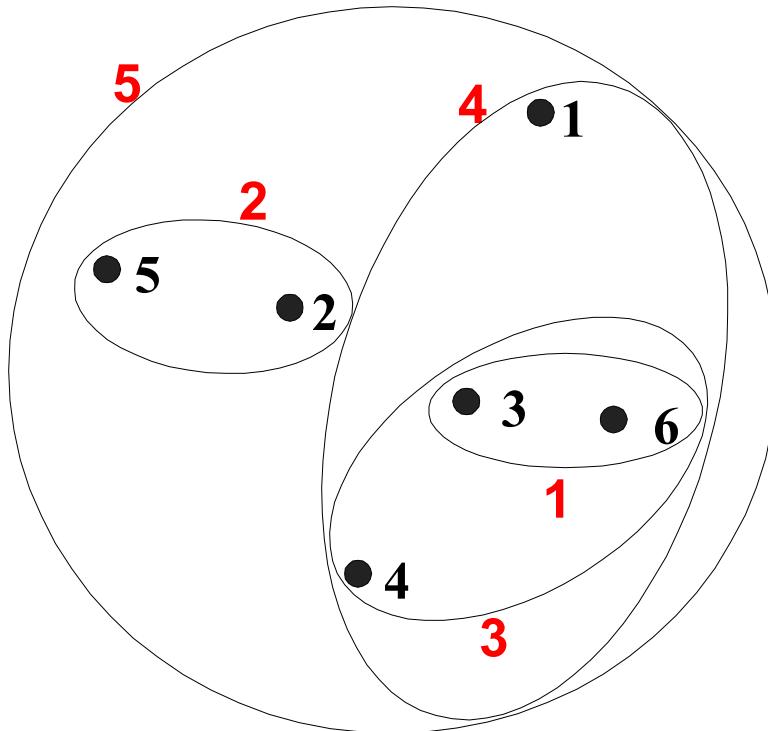
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

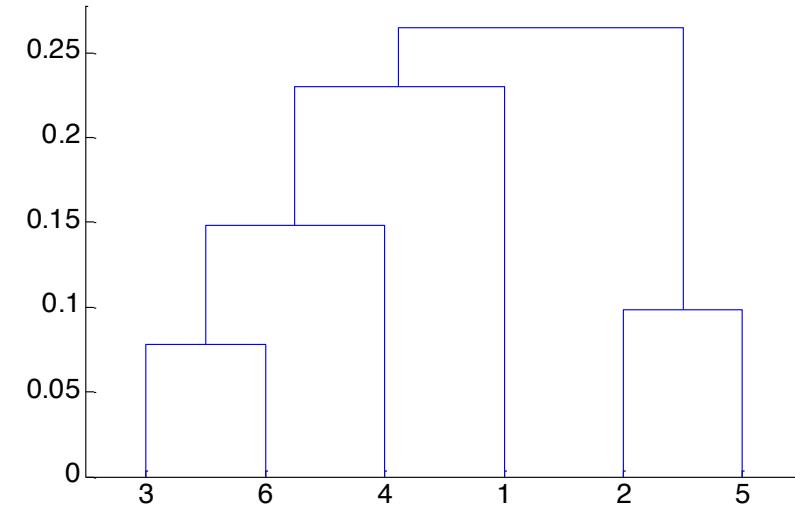
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

# Hierarchical Clustering: Group Average

---

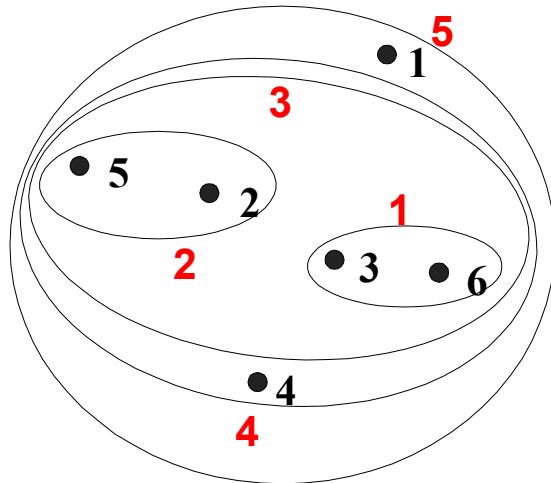
- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

# Cluster Similarity: Ward's Method

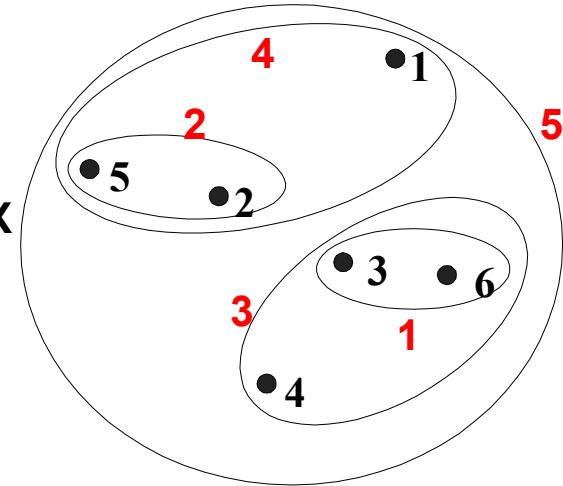
---

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

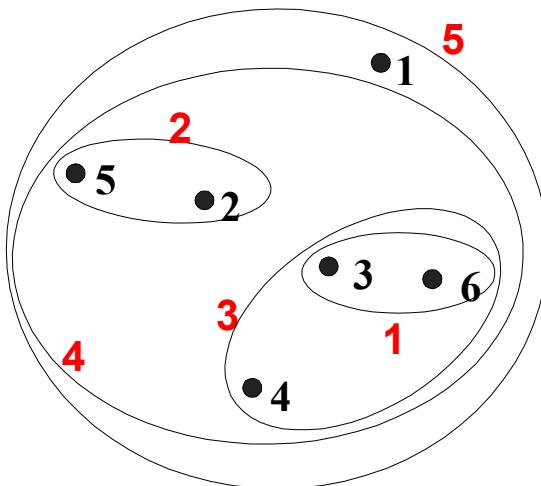
# Hierarchical Clustering: Comparison



MIN

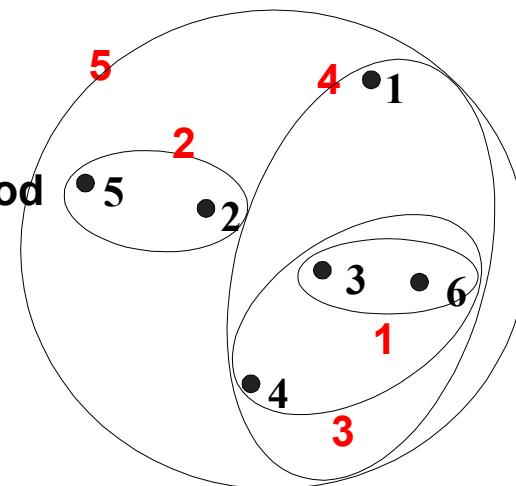


MAX



Group Average

Ward's Method



# Hierarchical Clustering: Time and Space requirements

---

- $O(N^2)$  space since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$  time in many cases
  - There are N steps and at each step the size,  $N^2$ , proximity matrix must be updated and searched
  - Complexity can be reduced to  $O(N^2 \log(N))$  time for some approaches

# Hierarchical Clustering: Problems and Limitations

---

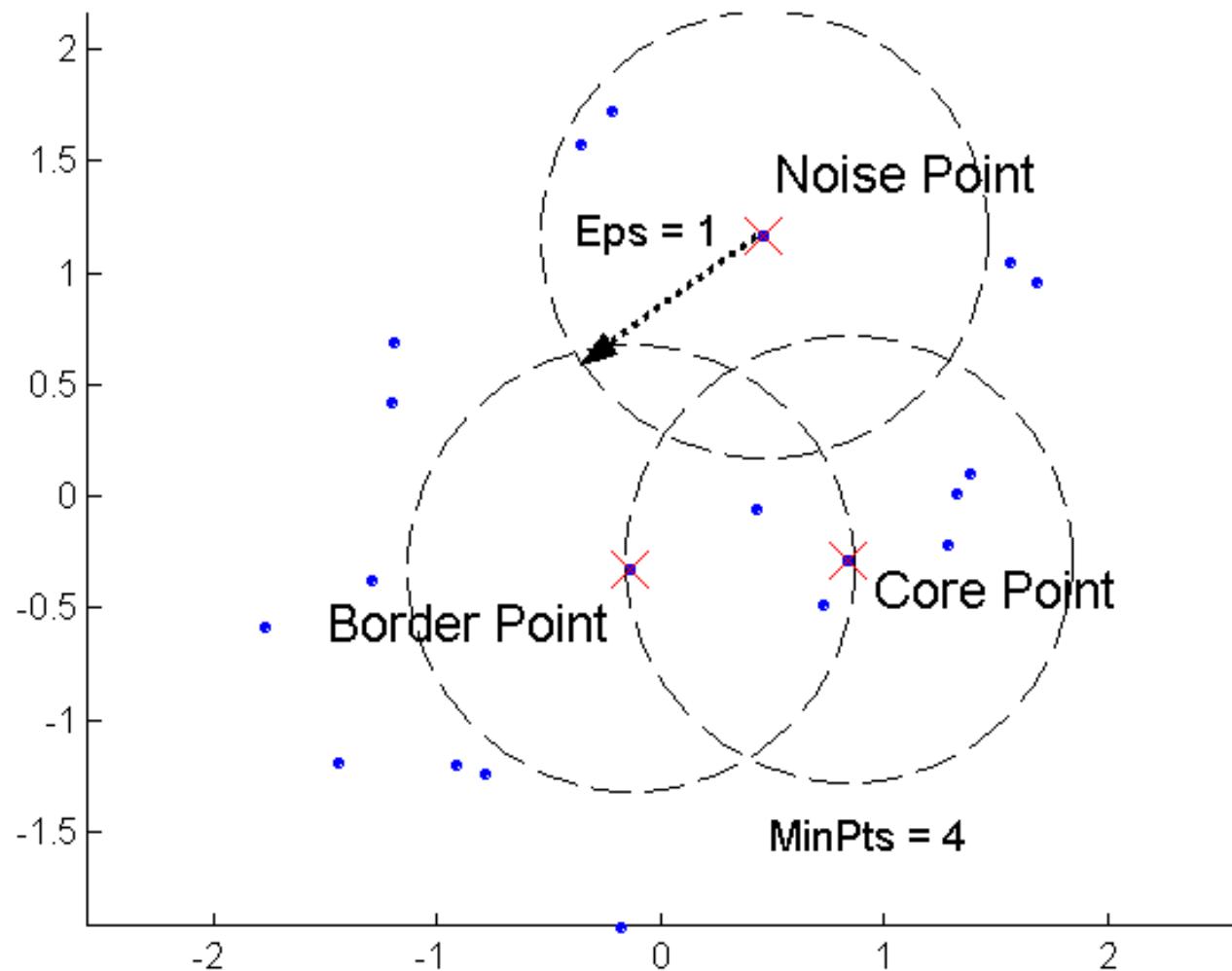
- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

# DBSCAN

---

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a **core point** if it has more than a specified number of points (MinPts) within Eps
    - ◆ These are points that are at the interior of a cluster
  - A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
  - A **noise point** is any point that is not a core point or a border point.

# DBSCAN: Core, Border, and Noise Points



# DBSCAN Algorithm

---

- Eliminate noise points
- Perform clustering on the remaining points

*current-cluster-label*  $\leftarrow 1$

**for** all core points **do**

**if** the core point has no cluster label **then**

*current-cluster-label*  $\leftarrow \text{current-cluster-label} + 1$

        Label the current core point with cluster label *current-cluster-label*

**end if**

**for** all points in the *Eps*-neighborhood, except *i<sup>th</sup>* the point itself **do**

**if** the point does not have a cluster label **then**

            Label the point with cluster label *current-cluster-label*

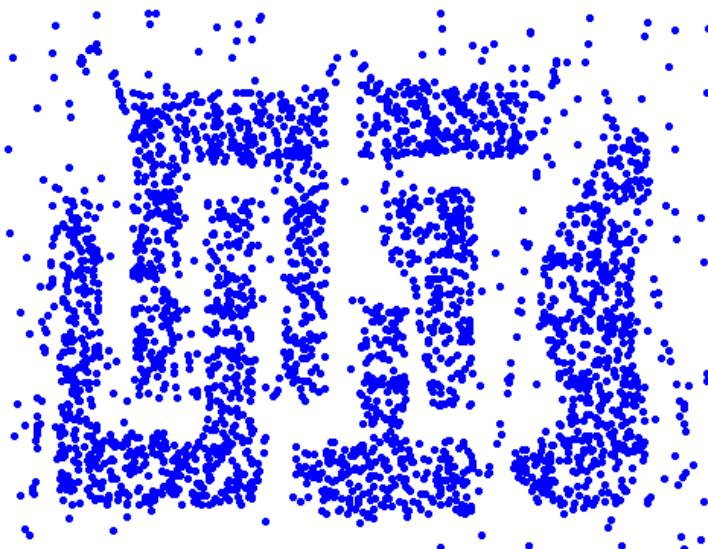
**end if**

**end for**

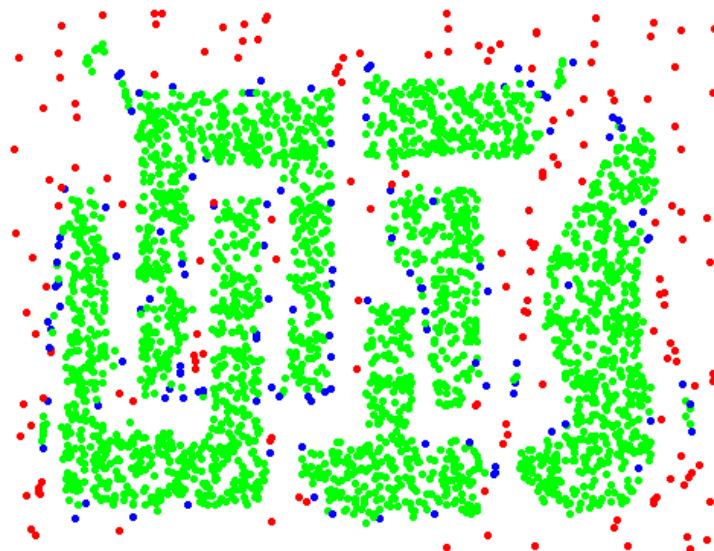
**end for**

# DBSCAN: Core, Border and Noise Points

---



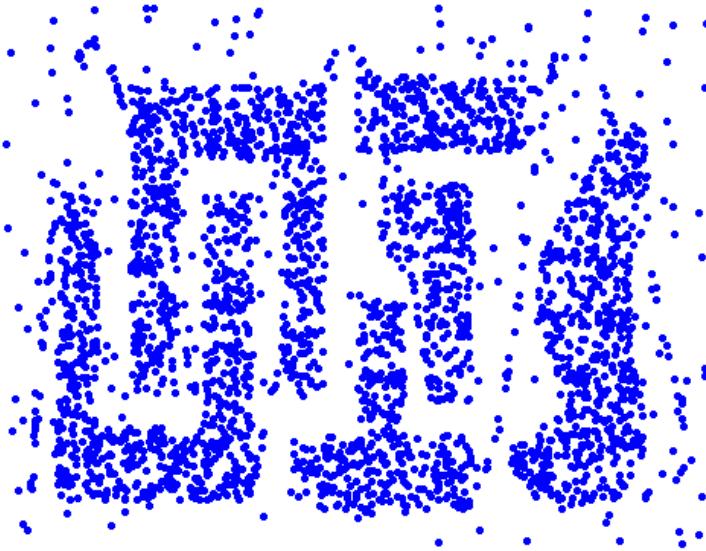
Original Points



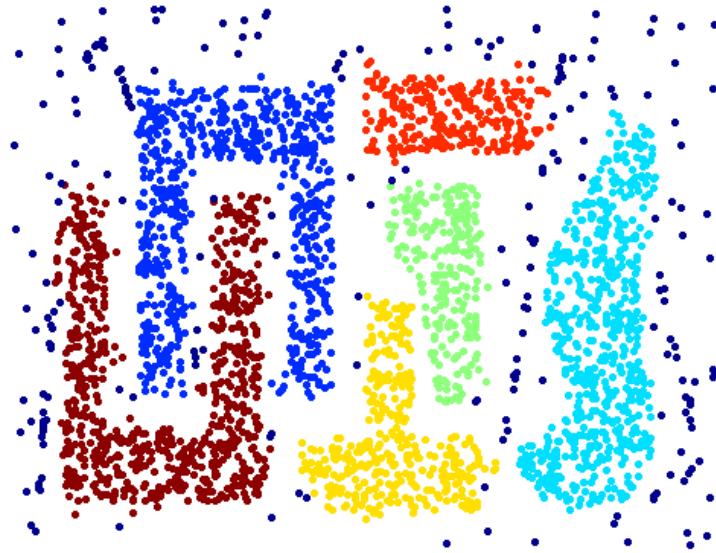
Point types: **core**,  
**border** and **noise**

Eps = 10, MinPts = 4

# When DBSCAN Works Well



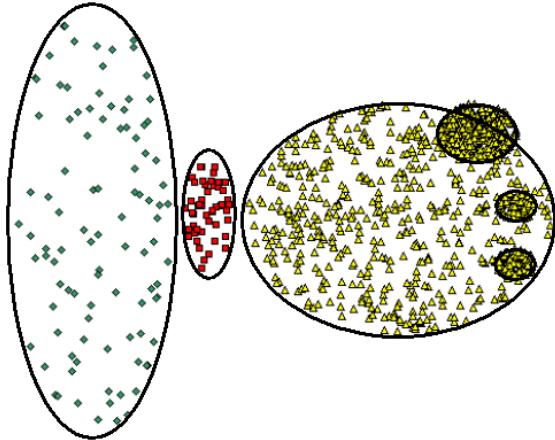
Original Points



Clusters

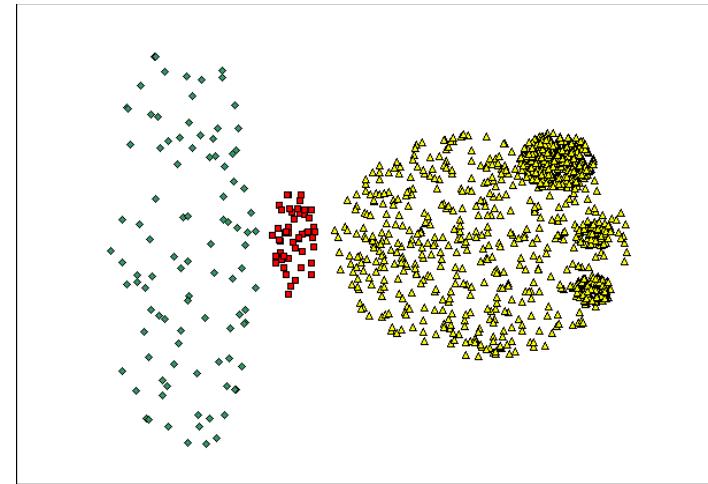
- Resistant to Noise
- Can handle clusters of different shapes and sizes

# When DBSCAN Does NOT Work Well

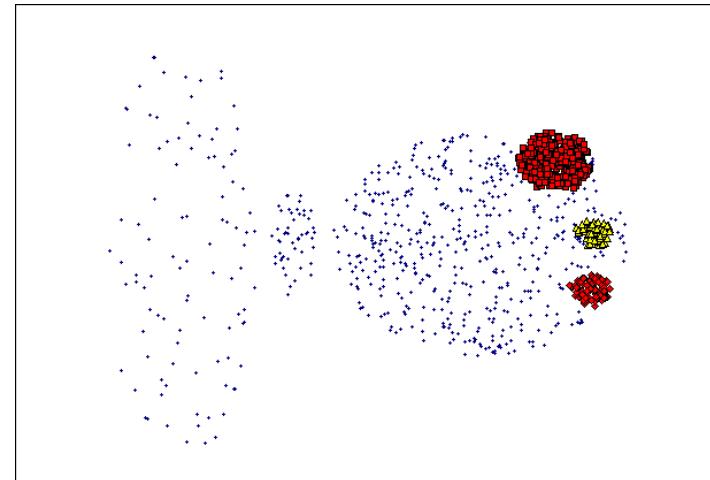


**Original Points**

- Varying densities
- High-dimensional data



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

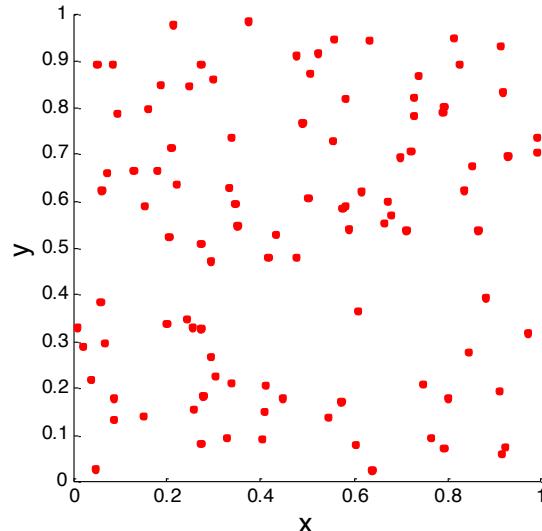
# Cluster Validity

---

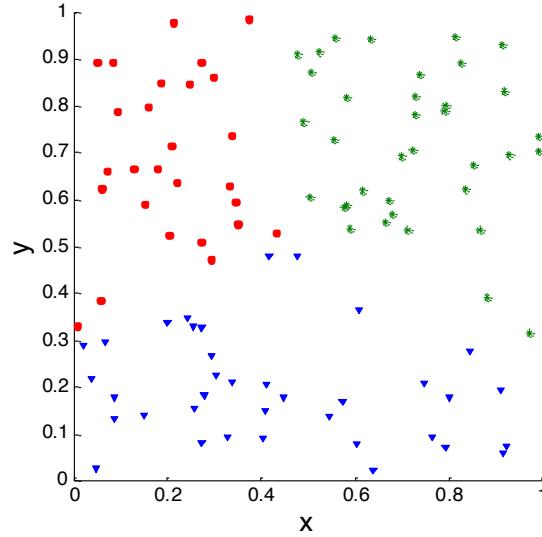
- For supervised classification we have a variety of measures to evaluate how good our model is
  - Accuracy, precision, recall
- For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?
- But “clusters are in the eye of the beholder”!
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

# Clusters found in Random Data

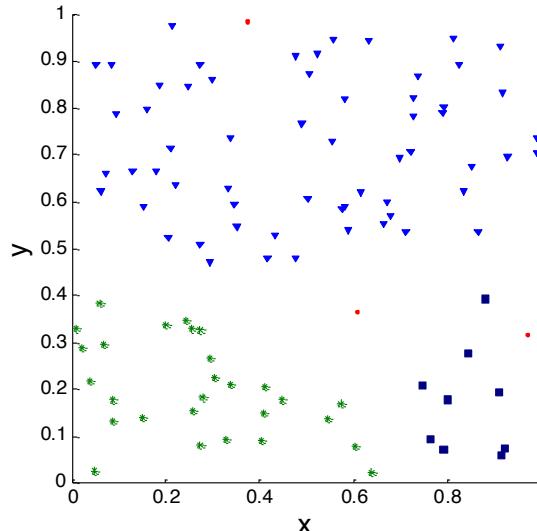
Random Points



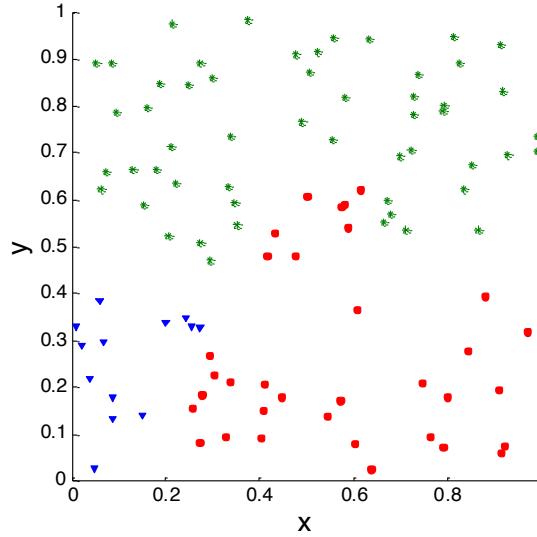
K-means



DBSCAN



Complete Link



# Different Aspects of Cluster Validation

---

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
  - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the ‘correct’ number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

# Measures of Cluster Validity

---

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - **External Index:** Used to measure the extent to which cluster labels match externally supplied class labels.
    - ◆ Entropy
  - **Internal Index:** Used to measure the goodness of a clustering structure *without* respect to external information.
    - ◆ Sum of Squared Error (SSE)
  - **Relative Index:** Used to compare two different clusterings or clusters.
    - ◆ Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as **criteria** instead of **indices**
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

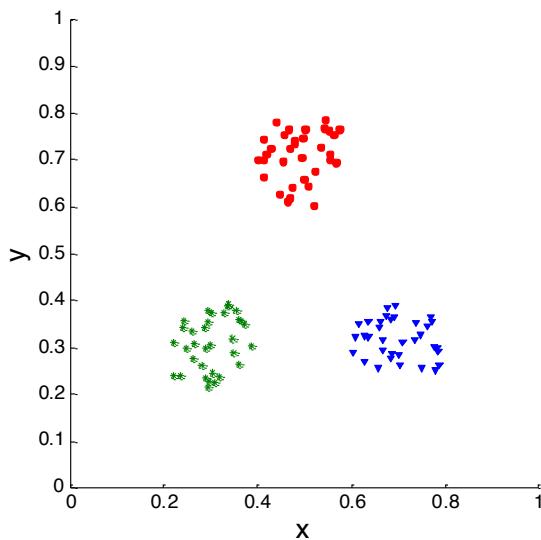
# Measuring Cluster Validity Via Correlation

---

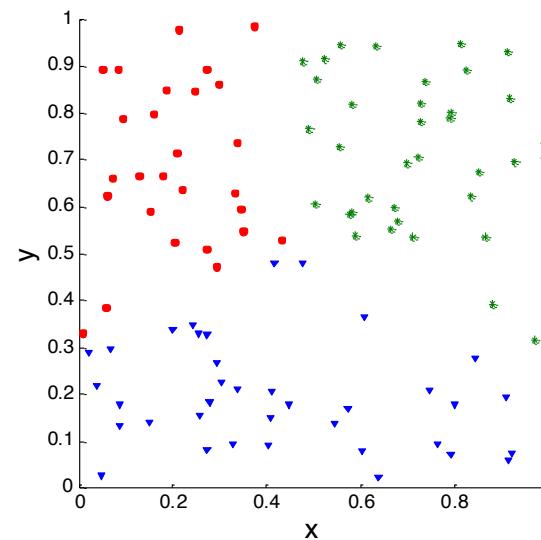
- Two matrices
  - Proximity Matrix
  - “Incidence” Matrix
    - ◆ One row and one column for each data point
    - ◆ An entry is 1 if the associated pair of points belong to the same cluster
    - ◆ An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between  $n(n-1) / 2$  entries needs to be calculated.
- High correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

# Measuring Cluster Validity Via Correlation

- Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.



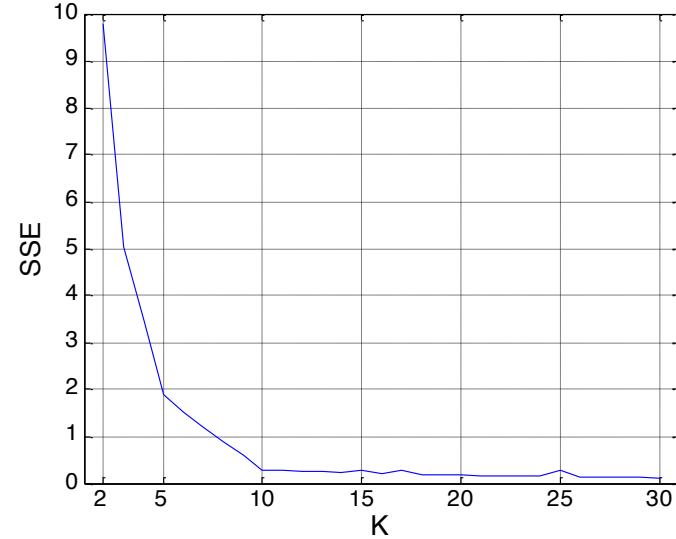
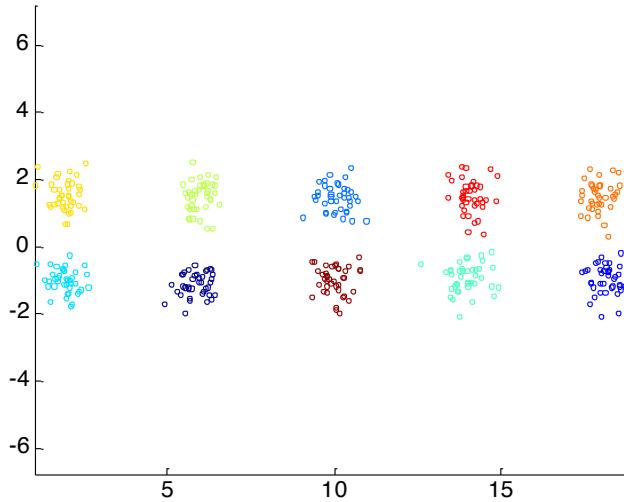
Corr = -0.9235



Corr = -0.5810

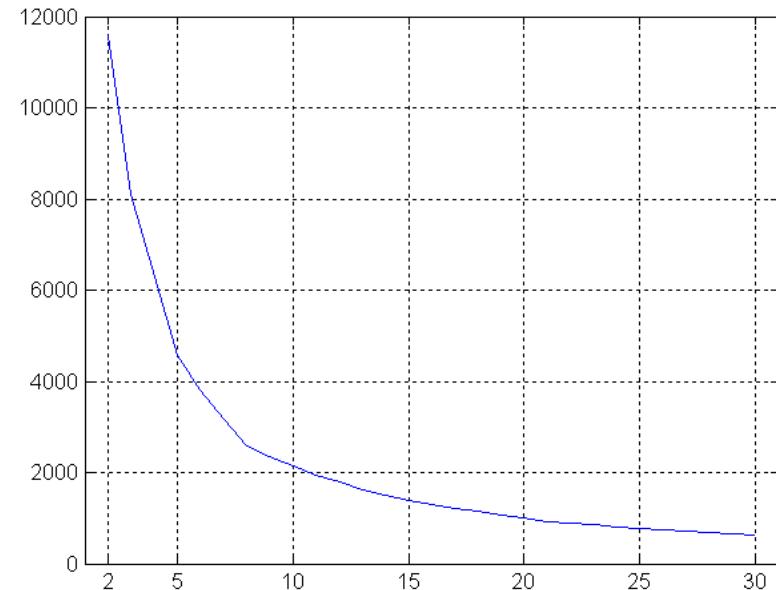
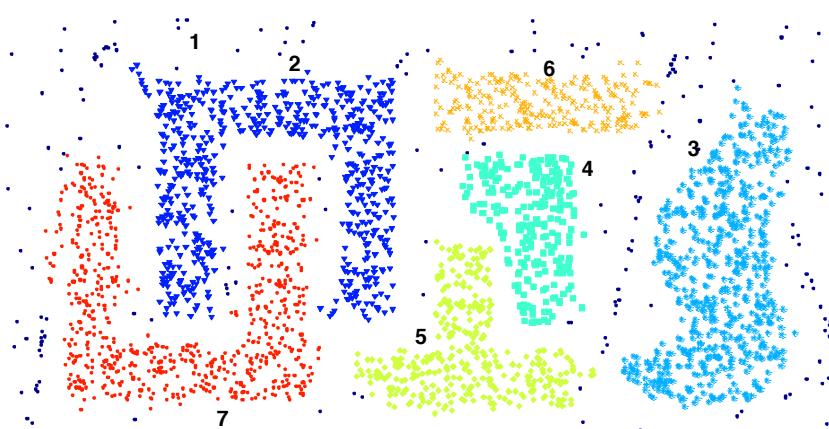
# Internal Measures: SSE

- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of clusters



# Internal Measures: SSE

- SSE curve for a more complicated data set



SSE of clusters found using K-means

# Framework for Cluster Validity

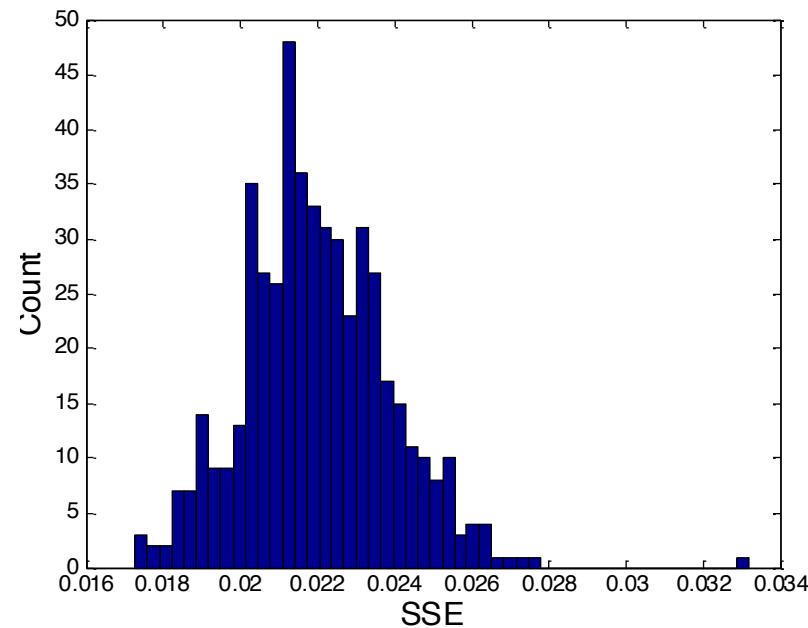
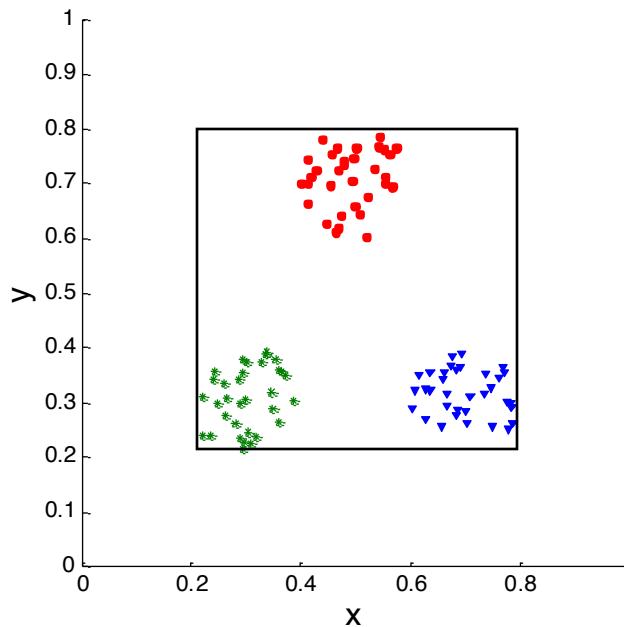
---

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - ◆ If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

# Statistical Framework for SSE

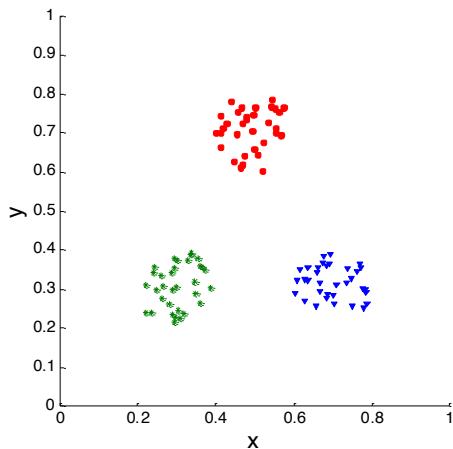
## ● Example

- Compare SSE of 0.005 against three clusters in random data
- Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

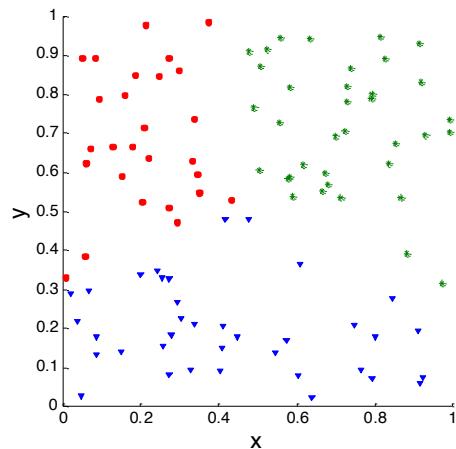


# Statistical Framework for Correlation

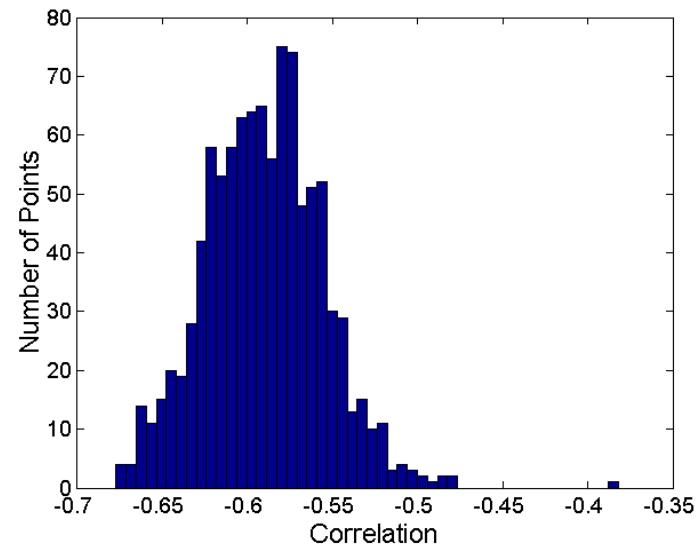
- Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.



**Corr = -0.9235**



**Corr = -0.5810**



# Internal Measures: Cohesion and Separation

---

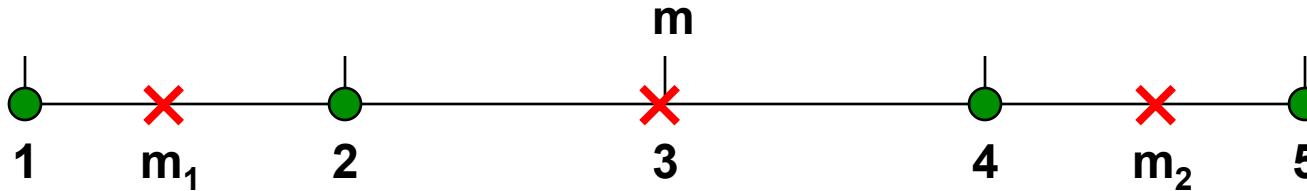
- **Cluster Cohesion:** Measures how closely related are objects in a cluster
  - Example: SSE
- **Cluster Separation:** Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)  
$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$
  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

– Where  $|C_i|$  is the size of cluster i

# Internal Measures: Cohesion and Separation

- Example: SSE
  - $BSS + WSS = \text{constant}$



**K=1 cluster:**

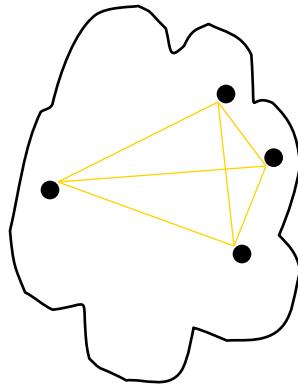
$$WSS = (1 - 3)^2 + (2 - 3)^2 + (4 - 3)^2 + (5 - 3)^2 = 10$$
$$BSS = 4 \times (3 - 3)^2 = 0$$
$$Total = 10 + 0 = 10$$

**K=2 clusters:**

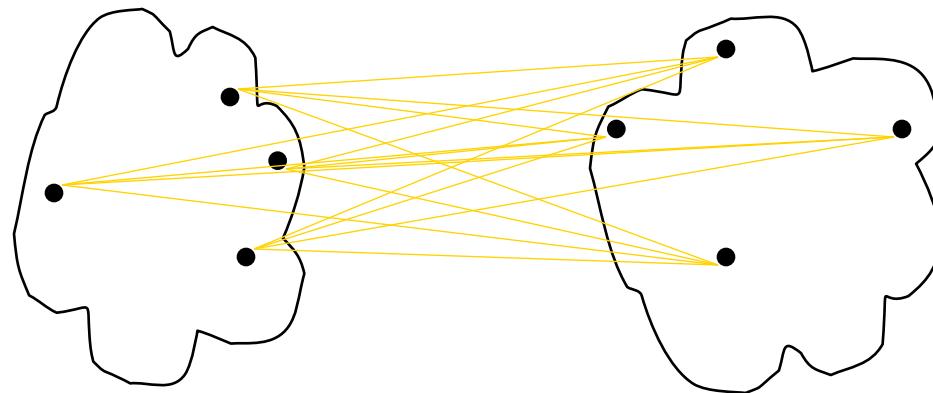
$$WSS = (1 - 1.5)^2 + (2 - 1.5)^2 + (4 - 4.5)^2 + (5 - 4.5)^2 = 1$$
$$BSS = 2 \times (3 - 1.5)^2 + 2 \times (4.5 - 3)^2 = 9$$
$$Total = 1 + 9 = 10$$

# Internal Measures: Cohesion and Separation

- A proximity graph based approach can also be used for cohesion and separation.
  - Cluster cohesion is the sum of the weight of all links within a cluster.
  - Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion



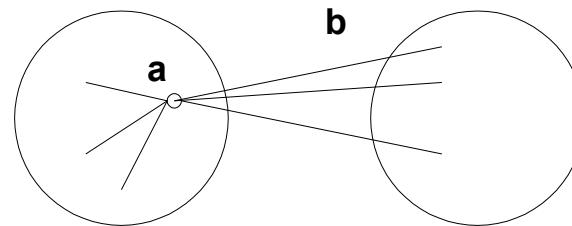
separation

# Internal Measures: Silhouette Coefficient

- Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings
- For an individual point,  $i$ 
  - Calculate  $a$  = average distance of  $i$  to the points in its cluster
  - Calculate  $b$  = min (average distance of  $i$  to points in another cluster)
  - The silhouette coefficient for a point is then given by

$$s = 1 - a/b \quad \text{if } a < b, \quad (\text{or } s = b/a - 1 \quad \text{if } a \geq b, \text{ not the usual case})$$

- Typically between 0 and 1.
- The closer to 1 the better.



- Can calculate the Average Silhouette width for a cluster or a clustering

# External Measures of Cluster Validity: Entropy and Purity

**Table 5.9.** K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

**entropy** For each cluster, the class distribution of the data is calculated first, i.e., for cluster  $j$  we compute  $p_{ij}$ , the ‘probability’ that a member of cluster  $j$  belongs to class  $i$  as follows:  $p_{ij} = m_{ij}/m_j$ , where  $m_j$  is the number of values in cluster  $j$  and  $m_{ij}$  is the number of values of class  $i$  in cluster  $j$ . Then using this class distribution, the entropy of each cluster  $j$  is calculated using the standard formula  $e_j = \sum_{i=1}^L p_{ij} \log_2 p_{ij}$ , where the  $L$  is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e.,  $e = \sum_{i=1}^K \frac{m_i}{m} e_j$ , where  $m_j$  is the size of cluster  $j$ ,  $K$  is the number of clusters, and  $m$  is the total number of data points.

**purity** Using the terminology derived for entropy, the purity of cluster  $j$ , is given by  $purity_j = \max p_{ij}$  and the overall purity of a clustering by  $purity = \sum_{i=1}^K \frac{m_i}{m} purity_j$ .

# Final Comment on Cluster Validity

---

“The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.”

*Algorithms for Clustering Data, Jain and Dubes*